uc3m | Universidad Carlos III de Madrid

Grado en Ingeniería Eléctrica
2017-2018

Trabajo Fin de Grado

"Sistema de medida de temperatura basado en NodeMCU y Android"

Teresa Castro Blanco

Tutor

Guillermo Robles Muñoz

Leganés a 25 de septiembre 2018

RESUMEN

El presente Trabajo de Fin de Grado pretende realizar una introducción al mundo del Internet de las Cosas, creando el diseño de un sistema de medida de temperatura, con conexión WiFi que permite a cualquier usuario con un dispositivo Android, acceder y obtener datos de temperatura del lugar donde se encuentre el dispositivo.

Es sistema de medida está formado por el microcontrolador ESP8266 y un sensor de temperatura. Para la visualización de la información se ha diseñado una aplicación móvil mediante la herramienta My App inventor 2.

Palabras clave

Internet de las Cosas, microcontrolador, ESP8866, Android, WiFi, sensor.

ABSTRACT

The main propose of this project is to introduce the reader into the world of the Internet of things and design and create a device that is able to measure temperature. This device includes WiFi connexion and allows anybody access to this information using an Android application.

The measuring system is based on an ESP8266 microcontroller and on a BME280 temperature sensor. Using My App Inventor, an application for Android systems has been designed for the visualization of the information.

Keywords

Internet of Things, microcontroller, ESP8266, Android, WiFi, sensor.

INDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

1.1. Introducción.

La aparición de las redes inalámbricas ha supuesto una gran evolución en la sociedad, ya que permite el acceso a información a la cual el individuo no acceso directo, en el momento que desea conocerla.

La sociedad pasa de tener acceso a la información desde punto fijo, como podía ser un ordenador con conexión a una red, a cualquier parte del mundo donde haya acceso a una red de datos.

Esta necesidad de conectarse a todo crea interés por establecer comunicaciones con todas las cosas que nos rodean. Así es como aparece el término "Internet de las Cosas", en inglés Internet of Things (IoT). Este término hace referencia a sistemas inalámbricos que conectan objetos físicos con el objetivo de generar una gran cantidad de información útil para mejorar cualquier aspecto del día a día, en cualquier lugar como podría ser en una empresa o en un domicilio.

Siendo conscientes o no, estamos en contacto directo con dispositivos que procesan una gran cantidad de información, como indica el informe Cisco del 2011. En el año 2010 la cantidad de dispositivos conectados a Internet superó al número de personas, y estima que para el año 2020, existirán 50 mil millones de dispositivos para 7,6 mil millones de personas, esto supone un promedio de 6,58 elementos por personas [1]. Indicativo de que actualmente la cantidad de proyectos basados en la conectividad de objetos es una necesidad en constante crecimiento.

Uno de los elementos clave para el desarrollo del Internet de las Cosas es incremento de creación de dispositivos electrónicos conocidos como microcontroladores, que permiten la conexión a Internet de sensores y actuadores para llevar a cabo acciones concretas [2].

El desarrollo de los microcontroladores con plataformas como Arduino ha permitido que expertos y principiantes puedan crear de sus propios proyectos. Teniendo grandes alternativas de bajo coste, diferentes métodos de comunicación y gran cantidad de sensores y actuadores. La tendencia del conjunto microcontrolador, sensor y actuadores es crear sistemas muy precisos y que los consumos de energía cada vez sean menores.

1.2. Objetivo del trabajo

El objetivo principal del presente Trabajo de Fin de Grado consiste en hacer una introducción a lo que actualmente conocemos como Internet de las Cosas y desarrollar un sistema de medida con control remoto y acceso a Internet. Para la elaboración de este sistema de medida se empleará el sensor BME280 que permite tomar medidas de temperatura, presión y humedad. La recopilación de datos se lleva a cabo mediante un microcontrolador que permite la conexión a una red WiFi, este es el módulo ESP8266 que se encuentra incorporado en una placa de desarrollo conocida como NodeMCU. Para

la visualización de la información se desarrollará una aplicación WiFi utilizando la herramienta My App Inventor 2 junto con una aplicación de Google, denominada Google Chart, que permitirá la creación de un gráfico para poder acceder a la información recogida mediante un dispositivo Android.

Los objetivos específicos de este trabajo son los siguientes:

- Comprender el funcionamiento del módulo ESP8266.
- Programación con el IDE[1] Arduino.
- Elaboración de una aplicación mediante My App Inventor.
- Establecer comunicación entre NodeMCU y el sensor.
- Establecer comunicación entre NodeMCU y una red WiFi.
- Establecer comunicación entre la aplicación diseñada y el WiFi.

1.3. Marco regulador

Actualmente no existe una regulación específica aplicable a proyectos basados en el Internet de las Cosas. Esta normativa está en actual desarrollo, en Estados Unidos, a principios de agosto de 2017 se presentó la propuesta de ley S.1961-Internet of Things (IoT) Cybersecurity Improvement Act of 2017 para establecer los estándares mínimos de seguridad para este tipo de dispositivos. La normativa para Europa se espera que aparezca pronto y no se diferenciará mucho de lo establecido por EE. UU., ya que eso permitirá unificar criterios.

El Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) ha creado un grupo de trabajo para la estandarización del IoT, para especificar aspectos de seguridad que se deben emplear, funcionalidades relativas al descubrimiento y reconocimiento de los dispositivos, autentificación de usuarios, alertas o gestión de la privacidad [3].

No existe en este momento una normativa específica para el Internet de las Cosas, pero si existe una regulación para las comunicaciones inalámbricas esta normativa es la IEEE 802.11, normativa aplicable a las conexiones WiFi.

También se ha tenido en cuenta el índice de protección del contendor de los componentes, esto se encuentra detallado en la IEC 60529.

Por último, el sensor BME280 cumple los requisitos de las restricciones de uso de sustancias peligrosas en aparatos eléctricos y electrónicos propuesto por la comunidad europea de septiembre de 2011 [4].

[1] IDE Integrated development environment.

1.4. Estructura de la memoria

El documento se encuentra dividido en diez partes distintas, ordenando los aspectos del proyecto de la siguiente manera:

- **Capítulo 2**, en este apartado se describen los principales elementos que han sido utilizados y métodos de comunicación del dispositivo desarrollado.
- **Capítulo 3,** en este apartado se lleva a cabo una descripción del módulo, principal que se utiliza en el trabajo.
- **Capítulo 4**, desarrollo del funcionamiento del sensor utilizado en el proyecto y de cómo llevar a cabo su puesta en marcha.
- **Capítulo 5,** en este capítulo se lleva a cabo una breve introducción del entorno de desarrollo Arduino como realizar la configuración inicial para el proyecto.
- **Capítulo 6**, explicación del funcionamiento del entorno de desarrollo de My App Inventor 2.
- **Capítulo 7,** se describe el funcionamiento de la herramienta Google Chart.
- **Capítulo 8,** en este capítulo se desarrolla una explicación de los códigos utilizados en cada uno de los entornos de programación que se han utilizado en presente proyecto.
- **Capítulo 9,** este capítulo se detallan los principales materiales y precios necesarios para la elaboración del proyecto y la planificación de este.
- **Capítulo 10,** presenta en este último apartado se realiza una conclusión y una explicación de posibles líneas de investigación.

2. ESTADO DEL ARTE

2.1. Internet de las Cosas

El Internet de las Cosas en inglés Internet of Things (IoT), es una expresión creada para hablar de la conectividad de cualquier objeto a Internet, de tal forma que es posible actuar sobre un sistema, sin la interacción directa de una persona. Este término fue introducido por el jefe de investigación de Xerex Weiser en 1998, aunque no será hasta 2009 cuando aparece este término públicamente, es en el RFID journal donde el profesor del MIT Kevin Ashton, hace referencia a esta expresión [5].

"Si tuviésemos ordenadores que fuesen capaces de saber todo lo que pudiese saberse de cualquier cosa-usando datos recolectados sin intervención humana- seríamos capaces de hacer seguimiento detallado de todo, y poder reducir de forma importante los costes y malos usos. Sabríamos cuándo las cosas necesitan ser reparadas, cambiadas o recuperadas, incluso si están frescas o pasadas de fecha. El Internet de las Cosas tiene el potencial de cambiar el mundo como ya hizo Internet. O incluso más." [6].

Por ello la conectividad es algo fundamental al hablar del Internet de las Cosas ya que el objetivo principal es que cualquier objeto pueda conectarse y ser accesible desde cualquier sitio y en cualquier momento. Estos objetos pueden ser sensores y actuadores. Los sensores envían la información que recogen para que posteriormente sea procesada y los actuadores funcionen según la situación del sistema en cada momento, esto permite que los sistemas funcionen autónomamente sin la necesidad de la actuación de las personas. La conexión que necesitan los sensores no es una conexión con gran ancho de banda, ni de gran potencia por ello el consumo de energía es muy bajo, tecnologías que permiten este tipo de comunicaciones son Bluetooth, WiFi, Zigbee, etc. [7].

Este desarrollo de la tecnología tiene un gran impacto en todos ámbitos del día a día como por ejemplo el transporte, la domótica, salud y empresas.

2.2. Tecnologías de comunicación

"Una red inalámbrica es aquella que usa ondas de radio para conectar los dispositivos, sin la necesidad de usar cables de ningún tipo" [8]. Las redes inalámbricas se usan principalmente para tener acceso a información desde lugares donde no se puede acceder. Estas permiten la conexión de dispositivos que no se encuentran en contacto directo, además las instalaciones inalámbricas son mucho más baratas e invasivas que las conexiones tradicionales, pero tienen varios inconvenientes como por ejemplo que las ondas electromagnéticas a través de las cuales se realiza la conexión son sensibles a sufrir interferencias y por ello deben estar reguladas en cada país, además es necesario tomar una serie de medidas para garantizar la privacidad de los usuarios.

Las redes inalámbricas se pueden agrupar en cuatro grupos según el alcance de la señal [9]:

1. **Red inalámbrica de área personal,** también conocidas como WPAN (Wireless Personal-Area Networks), basada en la norma IEEE 802.15. Tecnologías basadas en esta conexión son Bluetooth, IrDA, ZigBee y UWB. Son redes que se pueden utilizar para distancias de hasta unos 10 metros, es una conexión muy sencilla, eficiente y de muy bajo coste.
2. **Red inalámbrica de área local,** denominada WLAN (Wireless Local-Area Networks) este tipo de red proporciona un acceso inalámbrico de 100 metros. Dependiendo de que marca las comercialice se basan en una norma u otra, si lo hace con la empresa WiFi sigue el estándar de IEEE 802.11 mientras que si lo hace Hiperlan siguen el estándar de ETSI[2]. Esta última no ha tenido tanta repercusión ya que es mucho más compleja su conexión.
   El estándar IEEE 802.11 son un conjunto de especificaciones de control de acceso al medio y de la capa física, esquemas de codificación y transmisión para comunicaciones inalámbricas, para implementar redes inalámbricas de área local en bandas de frecuencia de 2 GHz, 5 GHz y 60 GHz [8].
   El estándar 802.11 define varios componentes principales, para presente trabajo necesitaremos las siguientes definiciones:
   - Estación, en inglés "Station" (STA), cualquier dispositivo que se conecta a una red WiFi.
   - Punto de acceso inalámbrico, en inglés "Access Point" (AP), también denominado estación base (BS), y es un dispositivo que permite el acceso a una red cableada a partir de WiFi.
3. **Red inalámbrica de área metropolitana,** o WMAN (Wireless Metropolitan-Area Networks), esta red es muy similar a la red anterior, las principales diferencias son su alcance ya que llega hasta los 50 kilómetros y la velocidad de transmisión de datos es mucho más rápida que la red WiFi. Esta conexión se basa en la norma IEEE 802.16.
4. **Red inalámbrica de área amplia,** también denominada WWAM (Wireless Wide-Area Networks), son aquellas redes que tienen un alcance de más de 50 kilómetros, se suelen usar para dar cobertura a grandes áreas como ciudades, países o sistemas de satélites. Para este tipo de redes es necesario usar frecuencias con licencia y principalmente hay dos tecnologías basadas en este tipo de conexión, la telefonía móvil y los satélites.
   - La red de **telefonía móvil** consiste en un área de cobertura dividida en celdas, en el centro de cada celda hay una estación base a la cual los dispositivos móviles se conectan. Estas estaciones base están conectadas a una central de conmutación de telefonía móvil que une el teléfono móvil y la red de telefonía.
   - Las **conexiones inalámbricas** se pueden realizar mediante satélites, debido a la gran área que permite cubrir y este tipo de comunicación puede ser muy útil para dar acceso a zonas donde no es posible acceder con otro

[2] ETSI, European Telecommunications Standards Institute.

tipo de conexión, pero para este caso sería necesarios dispositivos especiales.

2.3. Sistema electrónico

Un sistema electrónico, *figura 2.1*, es un conjunto formado por fuente de alimentación, sensores, actuadores y un circuito encargado de procesamiento y control del sistema [10].

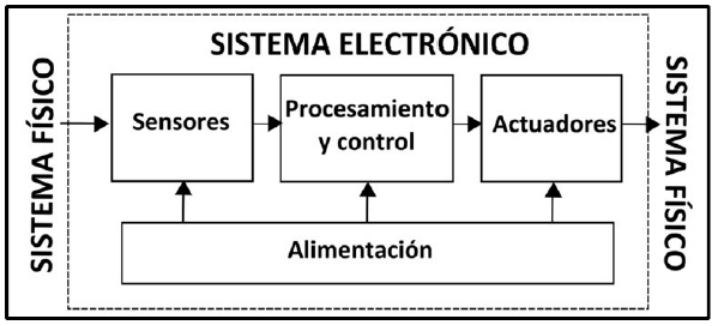


Figura 2.1.Sistema electrónico [10].

Los sensores obtienen la información de un sistema físico, y convierten esa información en una señal eléctrica que puede ser procesada por los circuitos internos.

Los circuitos internos de del sistema electrónico procesan esta señal dependiendo de las instrucciones que el hardware tenga grabado.

Una vez procesada esta señal los actuadores transforman la señal eléctrica en energía que actúa en un sistema físico.

Para que todo este proceso tenga lugar, todos los componentes del sistema necesitan recibir energía para llevar a cabo sus funciones y esta energía vendrá de una fuente de alimentación.

2.4. Sistema de medida

Un sistema de medida se puede considerar como un bloque donde la entrada es el valor verdadero y la salida es el valor medido. Si el sistema es perfecto la diferencia entre el valor de entrada y de salida sería nulo, pero esto en la realidad no ocurre ya que en toda medición existe un error de medida [11]. Las causas de este error de medida se deben principalmente a tres tipos de errores: error de precisión, errores sistemáticos, errores accidentales.

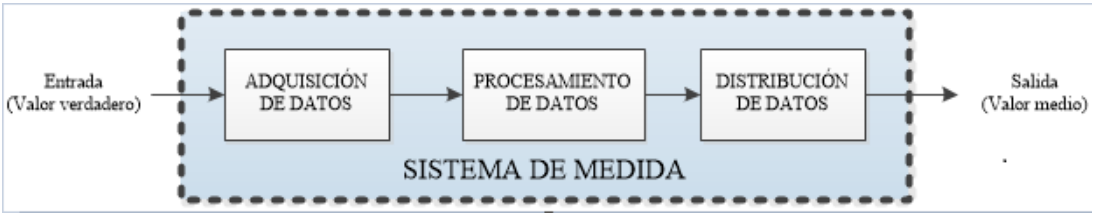


Figura 2.2. Esquema Sistema de medida electrónico [11]

En el sistema de medida podemos distinguir, como se muestra en la *figura 2.2* tres partes principales: adquisición de datos, procesamiento de datos y por último distribución de datos. La adquisición de datos es el proceso que se lleva a cabo cuando la variable a medir es obtenida y transformada a una señal eléctrica, esto puede realizarse mediante un sensor, en ocasiones la señal de entrada no se encuentra en las mejores condiciones para ser procesada y por eso se lleva a cabo un proceso de adecuación de esta, este proceso puede incluir acciones como: amplificación, aumento de la potencia de la señal, filtrado, para quitar las componentes no deseadas, linealización, obtención de la señal de salida solo en función de la entrada y por último modulación, es decir modificar la señal para disminuir la sensibilidad durante la transmisión de los datos. El procesamiento de datos consiste en el manejo de los datos para el uso definitivo de estos, este cometido lo lleva a cabo un procesador digital como un microcontrolador. La última etapa es el proceso de distribución de datos, los valores medidos pueden ser almacenados, mostrados en un display o transmitido a otro sistema [11].

Sensores.

Un sensor es un dispositivo capaz de obtener información del medio y transformar esa información en datos útiles para el receptor. Es decir, es un dispositivo electrónico capaz de transformar magnitudes físicas del entorno en magnitudes eléctricas o datos, como se muestra en la *figura 2.3* [11].

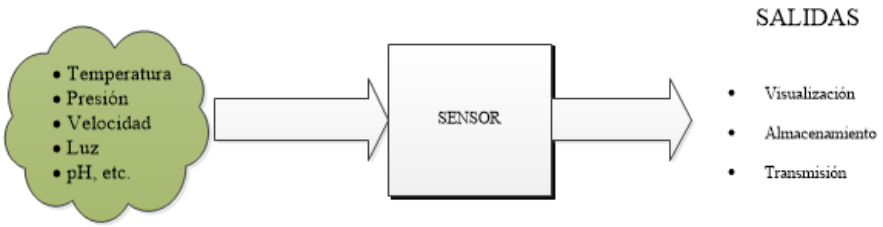


Figura 2.3. Sensor. [11]

Existe una gran cantidad de magnitudes físicas que pueden ser medidas como muestra la *tabla 2.1*:

TABLA 2.1. MAGNITUDES FÍSICAS Y VARIABLES DE MEDIDA. [11]

Naturaleza de la variable	Tipos de variables
Mecánica	Desplazamiento, velocidad, aceleración, fuerza, par, presión masa, flujo, etc.
Térmica	Temperatura, calor, entropía, etc.
Magnética	Campo magnético, flujo, permeabilidad magnética, etc.
Eléctrica	Carga, corriente, tensión, resistencia, conductancia, capacidad, permitividad dieléctrica, polarización, frecuencia, etc.
Óptica	Rayos gamma, rayos X, ultravioleta, visible, infrarrojo, microondas, etc.
Química	Humedad, pH, concentración iónica, análisis de gases, etc.
Biológica	Proteínas, Hormonas, antígenos, etc.

Los sensores se encuentran incorporados en el día a día de todas las personas, desde el sensor de la nevera que indica la temperatura a la que se encuentra el interior de esta, hasta nosotros mismos somos sensores ya que somos capaces de detectar humedad, temperatura, presión, luminosidad, etc. Existen numerosas clasificaciones de sensores, a continuación, se van a clasificar en cinco formas distintas [12]:

A. **Según su funcionamiento**. En esta clasificación podemos diferenciar en dos grupos:
   - Activos: son aquellos sensores que generan corriente o voltaje en respuesta a un estímulo del ambiente. Algunos ejemplos de este tipo de sensor son: células fotovoltaicas, acelerómetros, temporales, sensores de efecto Hall y piezoeléctricos. En la siguiente imagen se muestra un ejemplo de sensores activos.

Figura 2.4. Ejemplos de sensores de efecto Hall.

   - Pasivos: son los sensores que necesitan una fuente de energía de alimentación externa o auxiliar, algunos ejemplos de este tipo de sensores son: potenciómetros, galgas extensiométricas, termistores, RTD (detector de temperatura resistivo), magnetorresistivos e inductancias. En la siguiente figura se muestra un ejemplo de sensor pasivo.

Figura 2.5. Sensor RTD PT100.

B. **Según las señales de salida**. Según esta categoría tenemos en cuenta las señales, a la salida del sensor, estas señales pueden ser de dos tipos:
   - Digital: el sensor puede enviar únicamente dos valores, valor alto 1 y valor bajo 0, también puede mandar la información mediante un código de bits.
   - Analógicos: como su propio nombre indican envían una señal analógica de tensión o corriente, pudiendo tomar muchos valores entre un máximo y un mínimo.

   A continuación, se muestra un ejemplo de las diferentes señales que producen estos dos tipos de sensores.

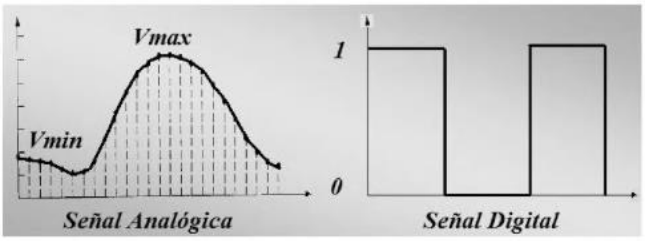


Figura 2.6. Ejemplos señales de sensores analógicos y digitales [12].

C. **Según la naturaleza de su funcionamiento.** En esta clasificación se realiza según la magnitud que se quiere detectar.
   - Posición. Los sensores de posición son aquellos que experimentan variaciones en función de la situación de sus partes.
   - Fotoeléctricos. Aquellos sensores que perciben cambios según la cantidad de luz que reciben.
   - Magnéticos. Estos sensores experimentan variaciones en el campo magnético que crean.
   - Temperatura. Sensores que perciben cambios en función de la temperatura en el medio en que se encuentren.
   - Humedad. Los sensores de humedad aquellos que detectan cambios en el nivel de humedad de la ubicación en la que se encuentren.
   - Presión. Aquellos sensores que reconocen cambios de la presión a la que se encuentran expuestos.
   - Movimiento. Estos sensores que distinguen variaciones de los diferentes movimientos a los que son sometidos.
   - Químico. Sensores que perciben cambios en los agentes químicos que alcanzan el sensor.

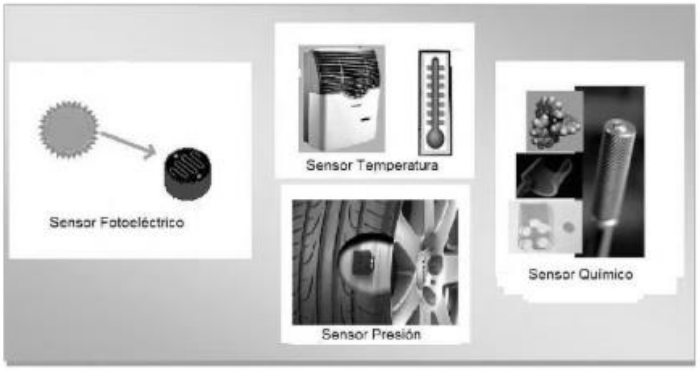


Figura 2.7. Ejemplos de sensores según clasificación de funcionamiento. [12]

D. **Según los elementos utilizados en la fabricación**. En esta clasificación se realiza según la naturaleza de funcionamiento del sensor.
   - Mecánicos. Sensores cuyos contactos abren o cierran.
   - Resistivos. Aquellos sensores que incluyen un elemento de carácter resistivo.
   - Capacitivos. Aquellos sensores que incluyen un elemento de carácter capacitivo.

- Inductivos. Aquellos sensores que incluyen un elemento de carácter inductivo.
- Semiconductores: Aquellos sensores que incluyen semiconductores.
- Piezoeléctricos: Sensores que utilizan cristales como el cuarzo.

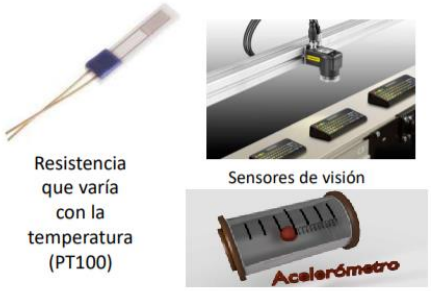


Figura 2.8. Ejemplos de sensores según elementos de fabricación.

E. **Según el nivel de integración**. En esta categoría se distinguen dependiendo de la conexión de los componentes y de las capacidades de tratamiento de datos.
   - Discretos. Son aquellos sensores en los que el circuito de acondicionamiento se realiza en componentes electrónicos discretos, separados e interconectados.
   - Integrados. En este caso el sensor y el circuito de accionamiento están implementados en un único circuito integrado.
   - Inteligentes. Son aquellos sensores que incluyen la capacidad de realizar algún cálculo numérico, comunicación, autocalibración, etc.

Para seleccionar un sensor para una aplicación determinada hay que tener en cuenta diferentes aspectos para alcanzar el mejor funcionamiento del sistema [12].

- Tecnología.
- Aplicación.
- Magnitud a medir.
- Ubicación de utilización.
- Coste.
- Características de entrada y salida.
- Posibilidad de amplificación.
- Posibilidad de control remoto.
- Multiplexación.
- Tamaño y peso.
- Vida media.
- Tensiones de alimentación y consumo de corriente.
- Características y especificaciones técnicas.

Un sensor tiene que transformar una magnitud física en una magnitud eléctrica, para que sea tratada correctamente, por ello en ocasiones hay que llevar a cabo una adecuación de la señal. Los sensores suelen incorporar estos elementos de adecuación de la señal para simplificar diseño, ahorrar espacio y disminuir costes [11].

Características estáticas y dinámicas de sensores

La actuación de un sensor se puede definir con la función de transferencia, ya que muestra el comportamiento estacionario, que es la relación entre la entrada y la salida cuando se alcanza el régimen permanente y el comportamiento dinámico indica el cambio que alcanza el calor final ante variaciones de la entrada del sistema [11].

Características estáticas:

Curva de calibración

La curva de calibración es la relación entre la magnitud física que se va a medir, variable de entrada "X" y la señal que se genera o variable de salida "Y" en régimen estático. Para poder definir esta curva es necesario definir los siguientes parámetros:

- Campo de medida, es el conjunto de valores que se encuentran entre los límites superior e inferior entre los que se puede realizar la medida.
- Alcance o fondo de escala, es la diferencia entre el límite superior e inferior que puede alcanzar la medida.
- Salida a fondo de escala, es la diferencia ente la salida del límite superior e inferior que puede alcanzar la medida.

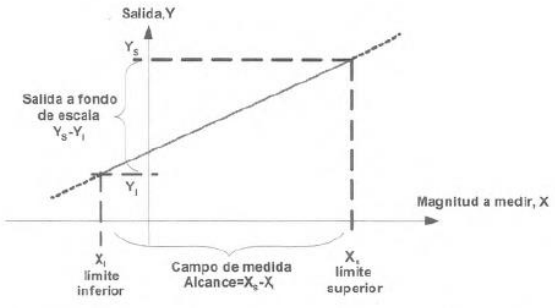


Figura 2.9. Ejemplo de campo de medida, alcance y salida a fondo de escala. [11].

En la *figura 2.9* se ejemplifica los conceptos de campo de medida alcance y salida a fondo de escala. Para definir la curva linealizada se emplean los siguientes términos.

- Sensibilidad, es la pendiente de la curva de calibración.
- No linealidad, es la máxima desviación de la curva de calibración con respecto a la función a la que se ha aproximado.

A continuación, la *figura 2.10* muestran dos ejemplos de linealización de una curva de calibración con diferentes valores de no linealidad.

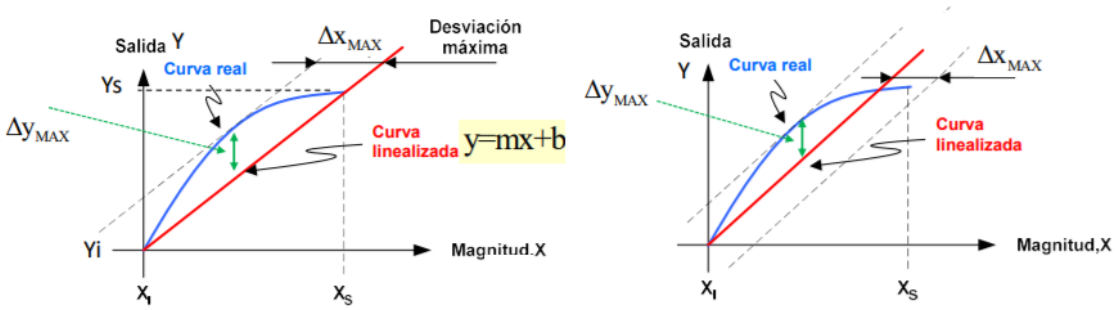


Figura 2.10. Ejemplos de linealización de una curva [11].

Cuando la curva de calibración no se puede linealizar, hay que definir los siguientes parámetros:

- Saturación, nivel de entrada a partir del cual la sensibilidad disminuye. En la siguiente *figura 2.11* se nuestra como a partir del valor marcado como X_{sat}, la curva de calibración empieza a saturar.

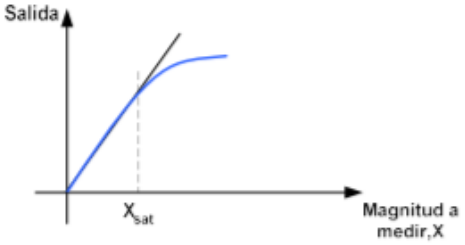


Figura 2.11. Curva de calibración saturada. [11]

- Histéresis, es la diferencia en la medida dependiendo del sentido en el que se realiza. Un ejemplo de esto lo podemos observar en la *figura 2.12.*

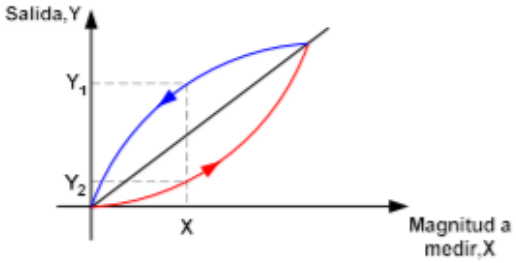


Figura 2.12. Curva de calibración que representa histéresis [11]

- Deriva es la variación de la curva de calibración respecto a un parámetro ambiental como por ejemplo la temperatura, presión o humedad, entre otras posibilidades.
- Zona muerta, es la zona de la curva de calibración que tienen una sensibilidad nula.
- Resolución el incremento mínimo necesario para que se pueda apreciar un cambio en la salida.

Errores

Al realizar cualquier medida, cometemos un error es imposible conocer magnitudes con exactitud absoluta, por ello siempre trabajamos con aproximaciones. Los principales términos para determinar el error cometido en una medición son: el error absoluto, que es la diferencia entre el valor medido y el valor exacto y el error relativo, que es el error absoluto dividido entre el valor exacto. Los errores también se pueden clasificar en sistemáticos y aleatorios. Los primeros se deben a defectos o fallos en el sistema de medida o a las condiciones ambientales. Los segundos son aquellos que no se puede controlar. Para determinar el error que se está cometiendo y comprobar de donde proviene la norma UNE 82009 [1.1-1.6] define los siguientes términos para cuantificar los errores [11].

- Veracidad, es el grado de concordancia entre el valor medio obtenido de un gran número de muestras y el valor verdadero que ha sido marcado como referencia. Se suele expresar en términos de sesgo o desviación.
- Precisión, es la capacidad de un sensor de obtener la misma salida cuando se realizan varias lecturas de la misma entrada en las mismas condiciones.
- Exactitud, es la capacidad de un sensor de proporcionar valores de medida cercanos al valor verdadero/ real / exacto / teórico de la magnitud a medir.

Cuando a partir de magnitudes medidas se calculan nuevos valores, estos se denominan medidas indirectas, y para calcular el error de estas medidas se lleva a cabo la teoría de propagación de errores donde el error de la medida indirecta depende del error de las magnitudes medidas y la operación en los valores medidos. [13]

Calibración

La calibración es el procedimiento para conseguir que los valores medidos se acerquen lo máximo posible a su valor real.

Características dinámicas

La respuesta de un sensor ante una señal de entrada variable en el tiempo, respuesta dinámica, es diferente a la respuesta ante una señal de entrada constante, respuesta estática. La respuesta dinámica de un sensor depende de cómo está constituido, y de los elementos que almacenan energía como las bobinas o condensadores. En la respuesta dinámica se produce una desviación entre entrada y salida y como el objetivo de cualquier elemento de medida es acercase lo máximo posible a la realidad, para conseguir esto existen unas herramientas matemáticas que permiten modelar los sistemas, estas herramientas son las Transformadas de Laplace o de Fourier. La mayoría de los sensores tienen un comportamiento dinámico comparable a un sistema de primer o segundo orden.

Las principales características de un sensor en el dominio de la frecuencia son:

- Velocidad de respuesta, es la capacidad de la señal de salida siga sin retraso las variaciones de la señal de entrada. Para definir este parámetro es necesario definir

otros tres: el tiempo de retardo, tiempo de subida y tiempo de establecimiento. El primero se trata del tiempo transcurrido desde la aplicación de un escalón de entrada hasta que la salida alcanza el 10% de su valor permanente se denomina como t_d. El segundo, tiempo de subida es el tiempo transcurrido desde el tiempo de retardo hasta que llega al 90% del valor permanente por primera vez, este valor se denomina como t_ry se suele utilizar para determinar el ancho de banda del sistema. En tercer lugar, el tiempo de establecimiento, se trata del tiempo que pasa desde la aplicación de un escalón de entrada hasta que la salida alanza el régimen permanente, se denomina como t_s. En la *figura 2.13,* se muestran los parámetros definidos anteriormente.

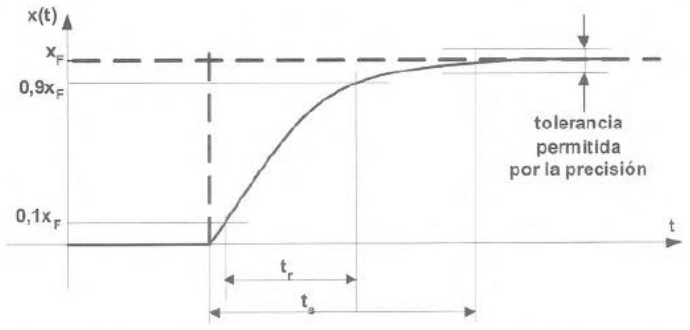


Figura 2.13. Respuesta sistema de primer orden ante entrada escalón [11].

- Respuesta frecuencial es la relación entre la sensibilidad y la frecuencia cuando la entrada es una sinusoide.
- Estabilidad es la deviación de salida del sensor al modificar parámetros independientes de los que se está midiendo.

Otras características importantes

Características de alimentación

La mayoría de los sensores necesitan de una fuente de alimentación para poder funcionar, hay que tener en cuenta los valores máximos y mínimos a lo que se puede alimentar cada sensor, la potencia consumida y disipada en ellos, y las impedancias de entrada y salida.

Características de aislamiento

Aquí hay que tener en cuenta la tensión de ruptura del sensor o de rigidez dieléctrica y la resistencia de aislamiento.

Características mecánicas

Hay que tener en cuenta el uso del sensor y tiene que tener un grado de protección adecuado para su utilización es decir dependiendo de donde vaya a situar debe tener una protección ante la aparición de fenómenos externos como pueden ser partículas, agua, el viento, descargas atmosféricas, vibraciones, impacto o contacto directo de las personas con partes peligrosas. Una mala elección puede producir que el sistema falle, o incluso puede suponer un riesgo para las personas. Para esto la Comisión Electrónica Internacional IEC define el índice de protección o IP (Ingress Protection), en la norma 60529.

2.5. Microcontrolador

Ya que nuestro trabajo consta de un microcontrolador Tensilica L 106, el cual incorpora el módulo ESP8266, expliquemos en primer lugar en que consiste un microcontrolador [10].

Un microcontrolador es un circuito integrado programable. Está formado por un conjunto de componentes conectados de una forma muy concreta. Siendo capaz de ejecutar una serie de instrucciones que han sido definidas con anterioridad.

Los tres componentes básicos del microcontrolador son:

- **CPU** (Unidad Central de Proceso). Se encarga de procesar las señales recibidas ejecutando cada instrucción definida previamente por el usuario y de controlar que la ejecución sea la correcta, dando lugar a unos datos de salida.

- **Memoria.** Aquí se almacenan todos los datos necesarios y las distintas instrucciones. Existen dos tipos de memoria:
  - Memoria volátil: La información almacenada en este tipo de memorias se pierde al desconectar la fuente de alimentación.
  - Memoria persistente: La información almacenada permanece incluso si se desconecta la fuente de alimentación.

- **Patillas de Entradas / salidas.** Estas patillas son las que permiten la comunicación con el microcontrolador con la parte física del sistema. En las patillas de entrada se conectan los sensores y en las de salida los actuadores. Hay que destacar que, muchas patillas, pueden ser empleadas como entradas o salidas indistintamente.

Como conclusión un microcontrolador es un circuito integrado encargado de la ejecución de las instrucciones previamente configuradas, por el usuario en su memoria, a partir de las señales eclécticas que recibe por las patillas de entrada y envía una respuesta por las patillas de salida.

2.6. Arduino

Arduino es un proyecto de hardware u software libre de ámbito mundial. Este proyecto nace de la necesidad de las aulas de obtener un dispositivo de bajo coste que funcionase en cualquier sistema operativo. Fue creada en 2005 en el Instituto de Diseño Interactivo de Ivrea (Italia) [13].

El concepto de Arduino engloba tres aspectos [6].

1. **Hardware libre.** Son placas de circuito impreso, cuyos ficheros esquemáticos de diseño están disponibles para que cualquier persona con los conocimientos adecuados y materiales pueda construirlo por su cuenta. Así todo el mundo puede

comprender su funcionamiento, modificarlo, reutilizarlos, mejorarlo y compartir los cambios.

2. **Software gratis, libre y multiplataforma.** Es decir, un entorno de desarrollo al alcance de cualquier usuario y está disponible para Linux, OS X y Windows. Este software lo tenemos que instalar en nuestro ordenador y mediante un cable USB podemos enviar cualquier conjunto de instrucciones que queremos que el microcontrolador realice. El conjunto de instrucciones es guardado en la memoria del microcontrolador, una vez cargada esta información no es necesario mantener conectado el microcontrolador al ordenador, ni es necesario volver a cargar el programa, al no ser que se realice una modificación, sino que sería suficiente con mantener la placa conectada a una fuente de alimentación para que funcione de forma independiente. Este entorno de desarrollo diseñado específicamente para la plataforma nos permite escribir, verificar y guardar los distintos programas, que el usuario cree.
   Para considerar que el software es libre se deben cumplir las cuatro libertades esenciales [14]:
   - **Libertad 0.** Libertad de utilizar el programa con cualquier propósito.
   - **Libertad 1.** Libertad de como estudiar cómo funciona el programa, y poder modificarlo como el usuario desee, es imprescindible tener acceso al código fuente.
   - **Libertad 2.** Libertad de redistribuir copias.
   - **Libertad 3.** Libertad para distribuir copias de versiones modificadas.

3. **Lenguaje de programación.** Es un idioma artificial que tiene unas reglas sintácticas y se utiliza para dar las instrucciones al microcontrolador. Comparte características comunes con otros lenguajes de programación como por ejemplo los bucles de condicionales.

2.7. Protocolo de comunicación sensor y hardware

A la hora de transmitir información desde un elemento electrónico a otro se puede realizar la comunicación de dos formas diferentes, se pueden trasmitir los datos de forma paralela, o en serie. En el primer caso se realiza el envío de bits simultáneamente por cada uno de los canales. En el segundo, el envío de bits se realiza por un único canal uno a uno. El primero necesita de dos canales para el envío de información, mientras que el segundo solo de uno, lo que supone una disminución de la complejidad, coste y tamaño del circuito [10].

En la comunicación en serie se puede realizar una clasificación según la forma de envío de información entre el emisor y el receptor, si el envío de datos es sincronizado es una comunicación síncrona y si no, asíncrona.

En la comunicación asíncrona, el emisor y receptor al comenzar cada conexión tienen que decidir los siguientes parámetros: la velocidad de transferencia de datos, el sistema de aviso de llegada y/o final de mensaje y los bits extra que se van a intercambiar para comprobar el correcto funcionamiento de la comunicación. Muchos controladores disponen de hardware UART. Esta comunicación usa siempre una línea para la transmisión de datos "TX" y otra para la recepción "RX". Esta comunicación siempre funciona de la siguiente forma envía un bit de inicio, bits de datos y un bit de parada. Para realizar esta conexión es necesario conectar cuatro cables diferentes, alimentación, tierra, TX y RX.

En la comunicación síncrona existe una señal periódica compartida que indica cuando se puede enviar o recibir datos. Los protocolos más comunes son la comunicación I^2C y SPI.

La comunicación I^2C en inglés Inter-Integrated Circuit, también conocido como TWI, TWo-wlre en inglés dos cables. En una de las comunicaciones más utilizadas, para comunicaciones a menos de un metro de distancia y se caracteriza por la utilización de dos líneas para transmitir la información, una para transferir datos denominada SDA y otra para el envío de la señal reloj llamada SCL, además de estas dos conexiones es necesario la conexión de alimentación y de tierra.

La ventaja que supone la conexión I^2C frente a la UART es que en la comunicación I^2C se puede conectar más de un dispositivo a la vez, pudiendo funcionar varios receptores o emisores simultáneamente, lo único que hay que considerar es que cada dispositivo que se conecte debe marcarse con una única dirección para poder identificarlo, además se debe configurar como maestro (inicia la transmisión de datos) o como esclavo(envía la respuesta), aunque esta característica puede cambiar. En la *figura 2.14,* se muestra la conexión I^2C que para que funcione adecuadamente se tiene conectar una resistencia "pull-up".

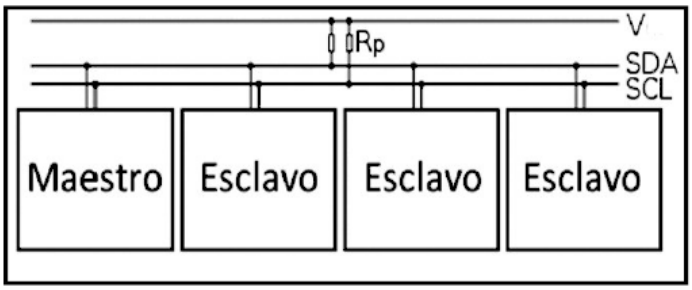


Figura 2.14. Esquema de comunicación I2C [6].

La comunicación SPI es un sistema de comunicación que permite controlar dispositivos electrónicos a distancias cortas, como ocurre en la comunicación I^2C hay que determinar que bus es maestro y cual esclavo. La principal diferencia es que este tipo de comunicación necesita seis conexiones distintas. La alimentación y tierra al igual que en los otros tipos de comunicaciones y la señal de reloj que se denomina "SCK, la línea "SS" que es la encargada de elegir qué dispositivo va a conectar de los distintos que puede

haber conectados, la línea "MOSI[3]" encargada del envío de datos y por último la denominada "MISO[4]" para el envío de datos en sentido contrario. En la *figura 2.15* se muestra un ejemplo de conexión de tres elementos con comunicación SPI.

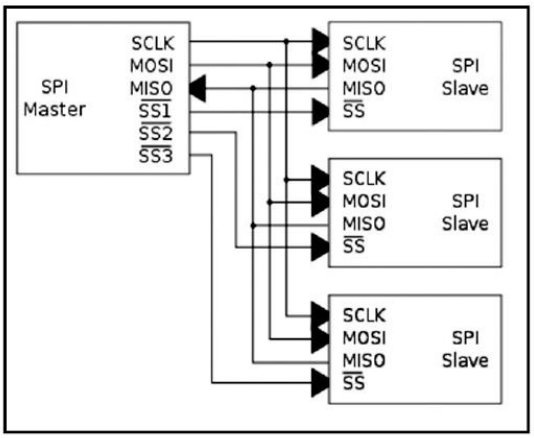


Figura 2.15. Esquema de comunicación SPI de tres elementos. [10].

Comparativa entre comunicación I^2C y SPI

La primera ventaja que podemos destacar de la comunicación I^2C es que utiliza menos pines para realizar la comunicación, estamos hablando, como se ha mencionado anteriormente, de dos cables para la comunicación y dos para la alimentación frente a seis cables de la comunicación SPI, dos para la alimentación y cuatro para la comunicación.

Otra de las ventajas de la comunicación I^2C, permite la conexión de varios dispositivos, lo único que habría que marcar claramente la dirección de cada uno de ellos.

Unas de las ventajas a destacar de la comunicación SPI es que consume menos energía que la I^2C y realiza una comunicación más rápida.

[3] MOSI (Multiple Output Single Input)

[4] MISO (Multiple Input Single Output.)

3. MÓDULO ESP82266

3.1. ESP-01 y ESP8266

Los primeros módulos creados para conectar a WiFi fueron elaborados por la empresa Espressif y fue el módulo ESP-01, al principio era muy complicado su uso con Arduino, pero ha ido evolucionando y este módulo es posible programarlo mediante Arduino, lo que simplifica mucho la programación. El módulo ESP8266 es un microcontrolador y puede realizar las mismas funciones que Arduino UNO [15].

Estructura ESP-01

Las partes de las que consta el módulo ESP-01 se muestran la *figura 3.16* [15]:

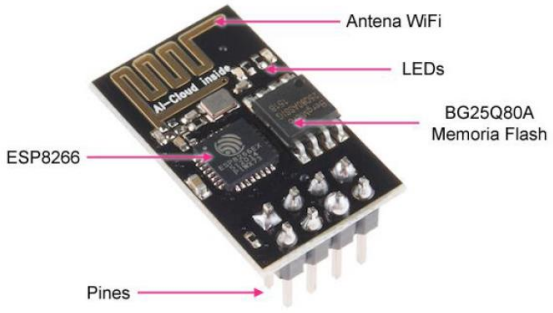


Figura 3.16. ESP-01 [15].

1. Antena WiFi permite la conexión a una red / Internet.
2. Led: indicador del funcionamiento del módulo.
3. Memoria flash, ya que el módulo ESP8266 no dispone de memoria donde cargar los programas.
4. ESP8266 microcontrolador del módulo ESP-01. Las principales características que incorpora el módulo ESP8266 son:
    - Contiene un chip Tensilica L106 de 32 -bit de bajo consumo.
    - Módulo WiFi de 2.4 GHz.
    - RAM, de 50 kB.
    - 1 entrada analógica de 10 bit (ADC).
    - 17 pines de entrada y salida GPIO.
    - El inconveniente de este módulo es que no posee una memoria Flash por ello habría que utilizar parte de los pines para conectarse a una memora externa.

En la siguiente figura se muestra el esquema del módulo ESP82266.

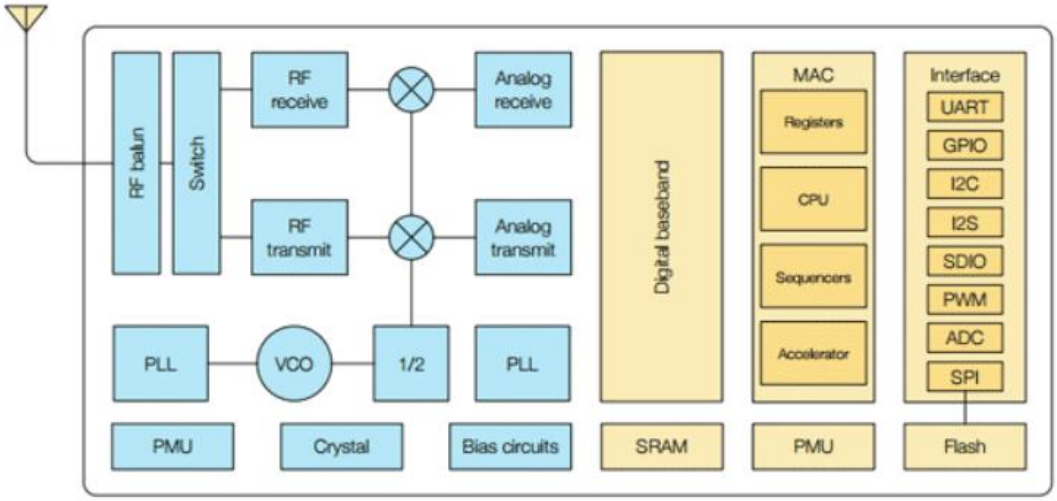


Figura 3.17. Esquema ESP8266 [16].

El firmware es el software de bajo nivel que permite controlar los circuitos eléctricos. El módulo ESP-01, viene por defecto instalado con una versión que permite comunicar con ESP8266 mediante comandos AT, usando el puerto serie. Existen otros firmwares que permiten controlar el microcontrolador ESP8266, como, por ejemplo: microPython, Espruino, ESPBasic y Arduino (para usar esta última opción solo es necesario crear el Sketch y cargarlo).

Esquema de entradas y salidas ESP-01

Una de las limitaciones del ESP-01 son los pines, ya que solo consta de ocho y cada uno de ellos tiene una función específica, como se puede observar en la *figura 3.18* [15].

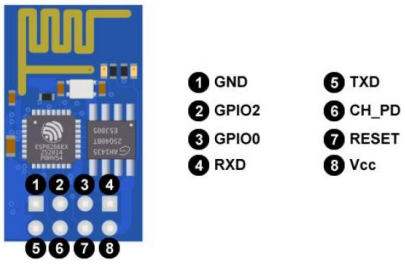


Figura 3.18.Esquema entradas y salidas ESP-01 [15].

1. GND: toma a tierra.
2. GPIO2: pin digital 2, se puede utilizar como entrada y salida.
3. GPIO0: pin digital 0, se puede utilizar como entrada y salida.
4. RXD: pin por el cual se reciben los datos del puerto serie, trabaja a 3.3 V. También se puede usar como pin digital, corresponde al GPI03.

5. TXD: pin por el cual se transmiten los datos del puerto serie, trabaja a 3.3 V. También se puede usar como pin digital, corresponde al GPI01.
6. CH_PD: pin para encender y apagar el módulo (0 V OFF y 3.3 V ON).
7. RESET: pin para resetear el módulo (0 V RESET).
8. Vcc: pin de alimentación del microcontrolador hay que alimentar a 3.3 V, con un máximo de 3.6 V y una corriente mayor a 200 mA.

Como se puede observar no hay ningún pin analógico y solo 4 pines digitales, y dos de ellos funcionan como pines E/S siempre y cuando el programa no los utilice para otras funciones, como por ejemplo mostrar información en el monitor serie.

El módulo ESP-01 soporta comunicación I^2C si los sensores se comunican mediante este protocolo no existe ningún problema, y tan solo con dos pines del ESP-01 podemos comunicarnos con infinidad de sensores.

Existe una amplia gama de módulos basados en el ESP8266, la principal diferencia entre ellos es el acceso a los pines. A continuación, en la *figura 3.19* se muestras los diferentes módulos basados en el ESP8266. Los pines del ESP8266 están cableados hasta los pines de ESP-12 para tener un acceso más sencillo, aunque sigue siendo complicado programar este tipo de módulos.

Figura 3.19. Diferentes módulos basados en ESP82266 [15].

Modos de funcionamiento de ESP8266

El módulo ESP8266 puede funcionar de seis formas distintas como se va a explicar a continuación [9]. En el apartado 2.2 se explicó lo que la norma IEE 802.11 define como estación y punto de acceso, conceptos importantes para desarrollar la explicación de las formas de funcionamiento.

1. **Estación y punto de acceso**. En esta configuración el módulo ESP8266 está conectado como estación permite acceder a una red WiFi, y como punto de acceso lo que permite crear redes malladas, como se puede ver en la *figura 3.20.*

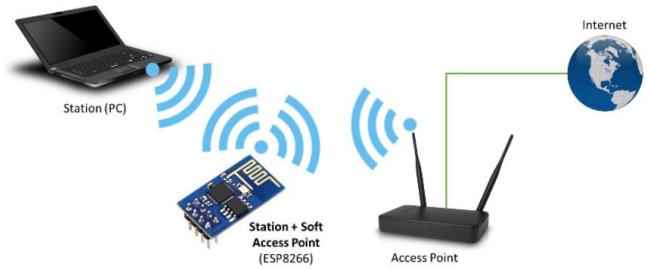


Figura 3.20. Esquema de funcionamiento como estación y punto de acceso [9].

2. Estación. Esta configuración se utiliza para conectar el módulo ESP8266 a una red WiFi mediante un punto de acceso como sería un rúter. Este modo de funcionamiento tiene una ventaja y es que en caso de que la comunicación falle automáticamente se volverá a conectar cuando esté disponible la conexión. El ejemplo de la conexión se muestra en la *figura 3.21.*

Figura 3.21. Esquema de funcionamiento como estación. [9].

3. Punto de acceso SoftAP. Este modo de funcionamiento se utiliza cuando queremos conectar dispositivos a una red WiFi y no conocemos la contraseña de esta. En primer lugar, los usuarios se conectan con el modo SoftAP y luego se proporcionan credenciales a la red y el módulo ESP8266 pasa a funcionar al modo estación. Podemos ver un ejemplo de este funcionamiento en la *figura 3.21*. El objetivo de este funcionamiento es crear redes malladas.

Figura 3.22. Esquema de funcionamiento como punto de acceso SoftAp [9].

4. Operando como un cliente, esto permite accedes a servicios proporcionados por servidores para que sea posible enviar, procesar y recibir datos. En la *figura 3.23*, se muestra el esquema de conexiones.

Figura 3.23. Esquema de funcionamiento operando como cliente [9].

5. Operando como un cliente seguro. Es la misma configuración que como cliente, pero incorpora un protocolo de comunicación seguro, se muestra un ejemplo de la conexión en la *figura 3.24*. El problema de este modo de funcionamiento es que solo es compatible con TLS[5] 1.1.

Figura 3.24. Esquema de funcionamiento como cliente seguro [9].

[5] TSL: seguridad de capa de transporte, es un protocolo de comunicación que proporcionan comunicaciones seguras por red.

6. Servidor este modo de funcionamiento se muestra en la *figura 3.14* y crea servidores que permite el acceso a clientes para enviar y recibir datos.

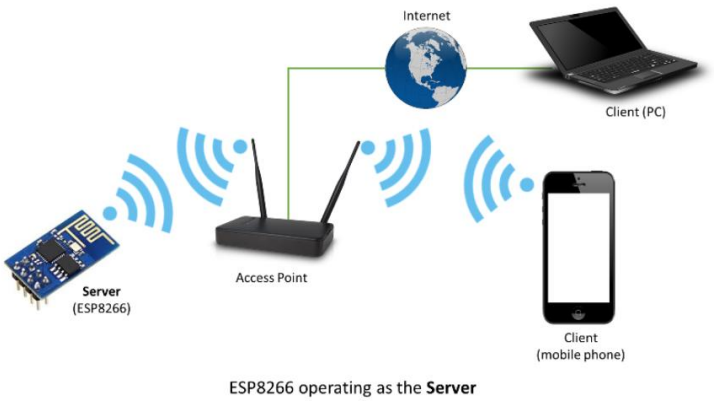


Figura 3.25. Esquema de funcionamiento como servidor [9].

3.2. Arduino Uno y ESP-01

Otra forma de programar cualquiera de los módulos ESP-xx mencionados en el apartado anterior es utilizando el módulo Arduino Uno como puente para cargar el programa, una vez cargado el módulo Arduino no es necesario para que el ESP-xx funcione, siempre y cuando se alimente con otra fuente.

En la *figura 3.26* se muestra un esquema de cómo, habría que llevar a cabo la conexión del módulo ESP-01 con el módulo Arduino UNO. Lo primero hay que conectar el reset del Arduino UNO a GND, esto lo que hace es que este módulo esté reseteando constantemente. Para alimentar el módulo ESP-01 se puede utilizar el pin de salida 3.3 V, pero esto supone un inconveniente y que la corriente suministrada por el microcontrolador Arduino Uno es de 50 mA y el módulo ESP-01 necesita de al menos 200mA, estaría funcionando, pero puede llegar a dar problemas, y fallar.

Otro problema a la hora de conectar el módulo ESP-01 con Arduino UNO es que los pines RX y TX de la placa Arduino UNO funcionan a 5 V, por lo que es necesario realizar un divisor de tensión para poder conectar los pines.

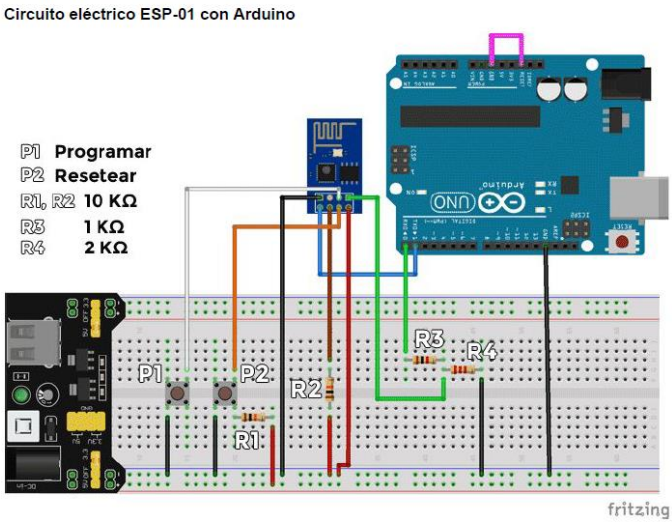


Figura 3.26.Esquema de conexión [15]

3.3. Placa de desarrollo NodeMCU

NodeMCU es al igual que Arduino una placa de desarrollo libre a nivel de software y de hardware. La principal ventaja que tiene este microcontrolador es que consta del microcontrolador ESP-12 o ESP-12E que incorpora como se ha mencionado anteriormente el módulo ESP8266 que permite la conexión WiFi, para elaborar proyectos de IoT con sistemas inalámbricos. La ventaja o diferencia del módulo ESP-12 frente al resto de módulos de basados en ESP8266 es como se encuentran cableados los pines, y la accesibilidad de estos.

Es importante en este punto destacar que NodeMCU no es un microcontrolador, sino que es una placa de desarrollo, cuyo objetivo es facilitar la programación de los componentes que la forman. El microcontrolador de esta unidad es el Tensilica L 106 y se encuentra integrado en el chip ESP8266 En la *figura 3.27*, se muestra un esquema de la estructura de la placa de desarrollo.

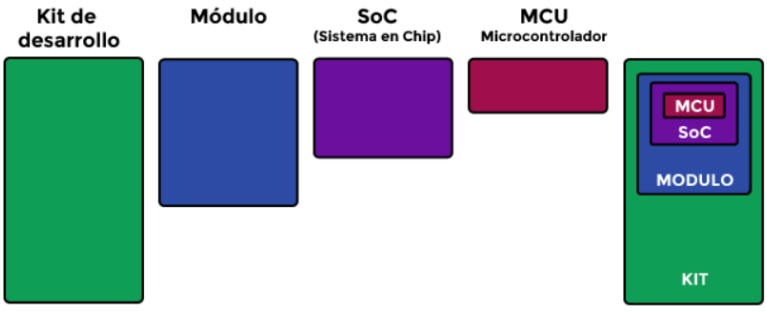


Figura 3.27. Esquema de la estructura NodeMCU. [15]

La placa de desarrollo NodeMCU será el que se encargue de las entradas, salidas y cálculos que requiera el programa en cada momento. Los principales fabricantes de este tipo de placas son: Amica (distribuidor oficial), DOIT y Loli /Wemos, en la *figura 3.28* se muestra un ejemplo de este tipo de módulos [15].

Figura 3.28. Módulo NodeMCU [15].

En la siguiente tabla se muestran las principales características del módulo NodeMCU.

TABLA 3.2. CARACTERÍSTICAS DE NODEMCU.

Procesador	Tensilica L106 32-bit
Tensión de operación	3.3 V
Frecuencia	80 MHz ~160 MHz
Pines E/S Digitales	12
Pines de E/S analógicos	1 (resolución 10 bits)
Memoria Flash	4 Mb
Memoria RAM	128KB
Conexión	WiFi
Soporte / Norma	802.11 b/g/n
Longitud	57 mm
Ancho	30 mm

Existen varias versiones de NodeMCU ya que es una placa de hardware libre y cualquier fabricante puede crear su modelo. Lo que hay que tener en cuenta es que todos los módulos se basan en el ESP866 y en el ESP-12 o ESP-12E, dependiendo de la versión con la que el usuario trabaje, este último componente es el que incorpora la memoria flash que el módulo ESP8266 no dispone. La principal diferencia entre las distintas versiones son los números de pines y las dimensiones [15].

Características generales de NodeMCU

- Conversor serie-USB para poder programar y alimentar a través de este.
- Fácil acceso a los pines.
- Pines de alimentación para sensores y componentes.
- Lees para indicar el estado del dispositivo.
- Botón reset.

Al comprar la placa es imprescindible reconocer la versión que tenemos ya que es importante conocer el esquema para poder hace la programación correctamente [17].

En la *figura* 3.*29*, se puede ver las tres versiones de NodeMCU y a continuación se explicarán las diferencias entre las versiones [15].

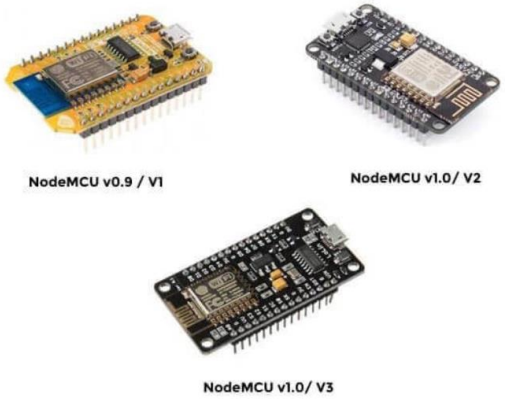


Figura 3.29. Versiones NodeMCU [15].

1ª Generación / V 0.9/ V1

Módulos que utiliza ESP-12 y ESP8266 con una memoria Flash de 4 MB. Tamaño 47 mm x 31 mm. El inconveniente de esta placa es el tamaño ya que ocupa todo el espacio de una protoboard y no deja hueco libre para conectar los pines.

2ª Generación/V 1.0 / V2

En este caso el chip que mota es el ESP-12E, y como ventaja que incluye respecto a su versión anterior es que su tamaño se ha reducido dejando una fila de pines disponible a cada lado en la protoboard para realizar las conexiones.

2ª Generación / V 1.0/ V3

Esta versión no es una nueva especificación oficial de NodeMCU, sino que es una versión creada por el fabricante Lolin. Las mejoras que incorpora un pin para la alimentación Vin a 5V, un GND adicional y un puerto USB más robusto. Los inconvenientes es que vuelve a ser más grande y no se puede acoplar a una protoboard.

Esquema de entradas y salidas NodeMCU

En la *figura 3.30* se puede observar el esquema de los diferentes pines a los cuales tenemos acceso en la placa NodeMCU, de la versión V3, que es la versión con la que se ha realizado el presente proyecto.

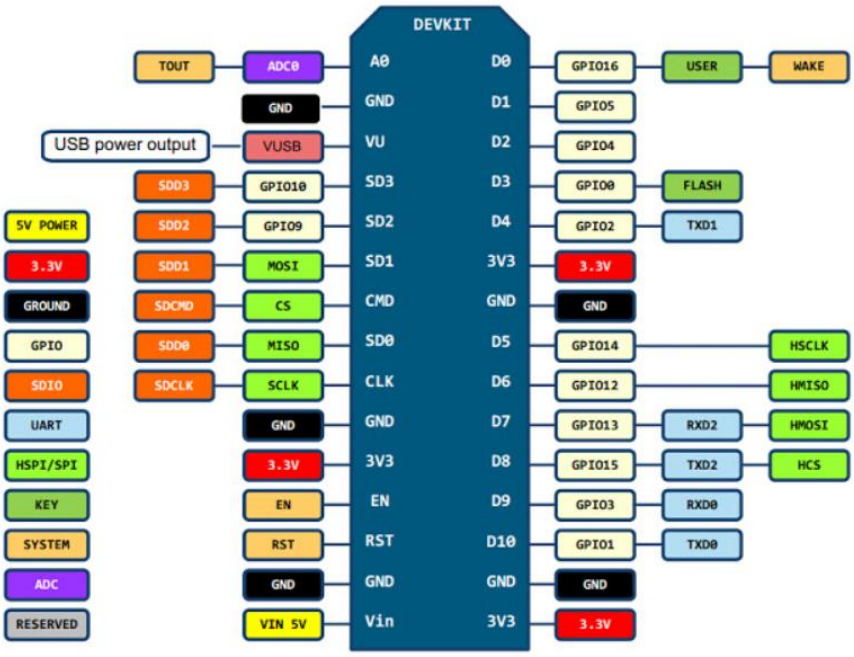


Figura 3.30. Pinout NodeMCU V3[6] [15].

Entradas y salidas digitales

NodeMCU, dispone de trece pines digitales numerados desde del D0 al D12. Está también indicado el nombre de los pines que conecta con el módulo ESP8266, la nomenclatura es la que aparece como GPIOxx. El módulo ESP8266 no dispone de memoria flash, por lo que hay que dejar pines libres para estas conexiones, por ello no es recomendable utilizar los, GPIO06, GPIO07, GPIO08, GPIO09, GPIO10 y GPIO11. El GPIO09 y GPIO10 corresponden respectivamente a los pines D11 y D12 hay que evitar el uso de estos pines, al igual que los pines D09 y D10 que corresponden a los pines Rx y Tx, usados para recibir y transmitir un programa o para la comunicación entre el NodeMCU y el ordenador a través del puerto serie. De este modo solo quedarían disponibles nueve pines digitales de entrada y salida, es decir desde el D0 al D8 [15].

Hay que tener en cuenta que al ser entradas y salidas digitales los sensores y actuadores que se conecten a estos pines deberían ser de tipo digital, ya que estas E/S solo están diseñadas para recibir y enviar valores binarios, es decir solo distinguen nivel alto y bajo.

[6] En el Anexo A, puede encontrar el esquema de pines de las tres versiones de NodeMCU

Entrada analógica

El NodeMCU cuenta con un pin analógico denominado A0. Este tiene un rango de valores de 0 a 3.3 V, como la placa solo puede trabajar con valores digitales la placa contiene un conversor analógico-digital, con una resolución de 10- bit, esto quiere decir que puede tomar 2^{10} (1024) valores diferentes, así que podrá distinguir valores entre 0 y 1023.

Salidas analógicas

A veces es necesario enviar señales analógicas como por ejemplo sería si quisiéramos controlar la intensidad con la que se enciende un led. Es decir, hay que enviar señal que varíe en el tiempo, señales analógicas. La placa NodeMCU al igual que otras como Arduino UNO no disponen de salidas analógicas como tal y este comportamiento lo podemos conseguir usando las salidas digitales de las que dispone el microcontrolador [10].

Para conseguir este efecto tendremos que crear una señal denominada PWM Pulse Width Modulation o Modulación de Ancho de Pulso). Esta seña lo que hace en lugar de emitir una señal continua, emite una señal cuadrada constituidas por pulsos de frecuencia constante, al variar la duración de los pulsos (nunca variando su frecuencia) se modifica la tensión promedio y con ello la variación de la tensión de salida. Lo que variamos es el ciclo de trabajo. Como se puede observar en la *figura 3.31*, cuanto menor duración de los pulsos menor valor promedio y cuanto mayor duración de pulso mayor tensión en la salida.

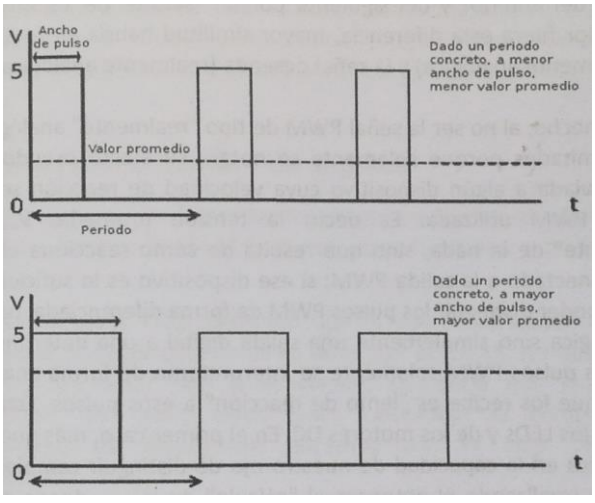


Figura 3.31. Tensión promedio con diferentes anchos de pulso [10]

A continuación, en la *figura 3.32* se muestra un ejemplo donde se obtienen la primera parte de lo que podría ser una se una onda sinusoidal, mediante la variación de ancho de pulso de la señal.

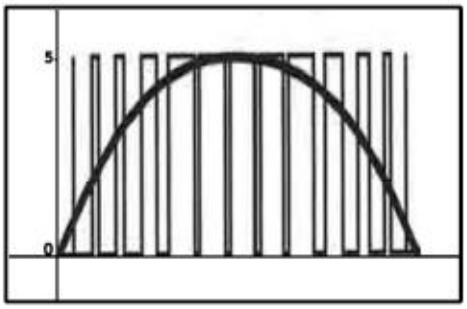


Figura 3.32. Variación de la tensión promedio con onda PWM. [10].

Alimentación

La placa NodeMCU dispone de pines de alimentación, estos pines tienen principalmente dos funciones, la primera alimentar la placa y la segunda alimentar sensores y componentes de salida. El voltaje de operación del dispositivo es de 3.3 V y por ello solo se podrán alimentar componentes a 3V. Al alimentar el NodeMCU con el puerto USB con 5 V, dispone de un regulador que saca 3.3 V para alimentar la placa y sacar a los pines para alimentar los componentes externos.

Existen dos formas diferentes de suministrar energía, y alimentar el microcontrolador de dos formas distintas:

- Mediante una fuente de alimentación como la que aparece en la *figura 2.8* de 3.3V y 5 V, usando la línea de potencia de 3.3 V.

Figura 3.33. Fuente de alimentación MB V2.

- Mediante un ordenador, el NodeMCU dispone de un conversor USB a serie, por ello con un simple cable USB, como el de la *figura 2.10*, tipo B conectamos a nuestro ordenador.

Figura 3.34. Cable USB 2.0 USB tipo B.

Leds y Botones

NodeMCU dispone de un led y dos botones. El led incorporado en el módulo está conectado al pin D04 y pertenece al módulo ESP-12E. Este led indica cuando se está cargando el programa, parpadea mientras esto ocurre además lo podemos programar y utilizar, pero hay que tener en cuenta que este led tiene la polaridad invertida, como se puede observar en la *figura 3.35* de la izquierda cuando en el código del D04 se encuentra en estado bajo el led conectado a la salida está apagado, mientras que en la placa está encendido y ocurre lo contrario cuando ponemos la salida a nivel alto, el led conectado a la patilla D04 está encendido mientras el led incorporado se apaga.

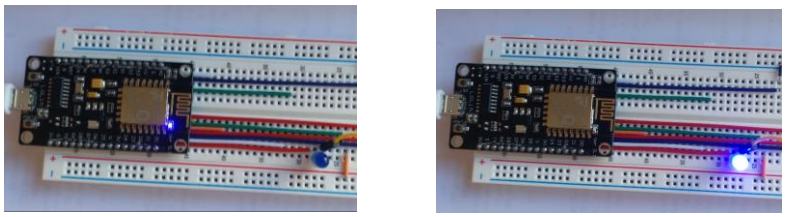


Figura 3.35. Ejemplo con NodeMCU.

Los dos botones que tiene la placa son el botón reset y flash. El botón reset, como su propio nombre indica es resetear. No elimina el código, lo que hace es iniciar la ejecución del programa desde el principio pasando por la función setup(). El botón flash, este está conectado al pin GPIO0 y corresponde con el pin D03. Lo que hace este botón en poner un pin en un estado conocido. Por ejemplo, el D3 se encuentra en estado normal en nivel alto es decir a 3.3 V y al pulsar el botón Flash se pone a 0 V o nivel bajo.

3.4. Comparación NodeMCU, Arduino UNO y ESP8266.

Como se ha podido observar utilizar el módulo NodeMCU es mucho más sencillo ya que tienen las conexiones hechas, y no hay que hacer divisiones de tensión ya que todo el módulo funciona a los 3.3 V que necesita el módulo ESP-12, lo contrario que ocurre con el módulo Arduino UNO.

Cabe destacar que realmente Arduino UNO estaría desaprovechado ya que solo se utilizaría para pasar la información del programa al módulo. Si el usuario realizase la conexión necesitaría: un Arduino UNO, que tiene un precio aproximado de unos 20,00 € y el módulo ESP-01 con un precio de 4,00 € aproximadamente, más resistencias con un precio entorno a los 2,00 €, cables para hacer las conexiones mencionadas anteriormente cuestan alrededor de 1,00€ y por último un cable para conectar Arduino UNO al ordenador, este es un cable USB estándar con conector macho tipo A y conector tipo B con un precio de 8,00 €, el total sería aproximadamente de unos 34,00 € y el módulo Arduino realmente no estaríamos usándolo frente a los 9,00 € que cuesta el módulo NodeMCU y las comodidades que esto supone se ha elegido desarrollar el proyecto con este dispositivo.

4. SENSOR BME280

4.1. Sensor BME280

El sensor BME280 es un sensor digital de temperatura, presión y humedad. Es un módulo con una precisión alta, bajo consumo energético y un formato muy compacto. Basado en tecnología BOSCH, un nivel de alta precisión, así como estabilidad a largo plazo. Con conexión I^2C o SPI. En el sensor de temperatura integrado está optimizado el ruido para obtener una resolución más alta. Para este proyecto solo se utilizará el sensor de temperatura, pero este sensor también puede tomar medidas de presión y humedad.

TABLA 4.3. ESPECIFICACIONES TÉCNICAS BME280. [4]

Magnitud	Valores
Voltaje de operación	1.71-3.6 V
Interfaz de comunicación	I^2C o SPI (3.3V)
Rango de Presión	300-1100 hPa
Resolución	0.16 Pa
Precisión absoluta	1 hPa
Rango de temperaturas	-40 ºC a 85 º C
Precisión de temperatura	1ºC
Rango de humedad relativa	0-100% RH
Precisión de HR	$\pm$3%
Consumo de energía	Bajo. sensor desactivado consumo 0.1 µA
Frecuencia de muestreo	157 Hz (máx.)
Dimensiones	2.5 x 2.5 mm
Altura	0.93mm

4.2. Conexión del sensor

En la *figura 4.36* se muestra las conexiones del sensor BME280. Vcc se encuentra conectado a una salida de 3.3 V del microcontrolador, GND a una salida GND del micro la salida del sensor marcada como SCL al pin D1 y por último SDA al pin D2.

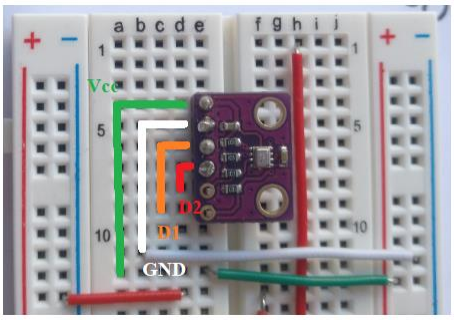


Figura 4.36. Esquema de conexión del sensor BME280.

En la *tabla 4.4* se muestra un resumen de las conexiones, para un protocolo de comunicación I^2C, que es el utilizado en este proyecto, mientras que en la *tabla 4.5* se muestra el conexionado para comunicación SPI

TABLA 4.4. RESUMEN CONEXIONES SENSOR BME280 COMUNICACIÓN I2C.

Pin del sensor BME280	Pin de NodeMCU
Vcc	3.3 V
GND	GND
SCL	D1
SDA	D2

TABLA 4.5. RESUMEN DE CONEXIONES SENSOR BME280, COMUNICACIÓN SPI

Pin del sensor BME280	Pin de NodeMCU
Vcc	3.3 V
GND	GND
SCL	D1
SDA	D2
CSB	D3
SD0	D4

5. ENTORNO DE PROGRAMACIÓN ARDUINO

5.1. IDE Arduino

En este apartado se realiza una breve introducción de las diferentes partes del software libre Arduino [10].

Para poder comenzar debemos descargar el software de Arduino, este se puede obtener la página oficial de Arduino. Una vez descargado y ejecutado al abrirlo aparecerá en nuestro ordenador la siguiente IDE (Integrated Development Environment), lo que se conoce como entorno de desarrollo integrado. Es la herramienta que permite escribir y editar el programa que ejecutara el microcontrolador.

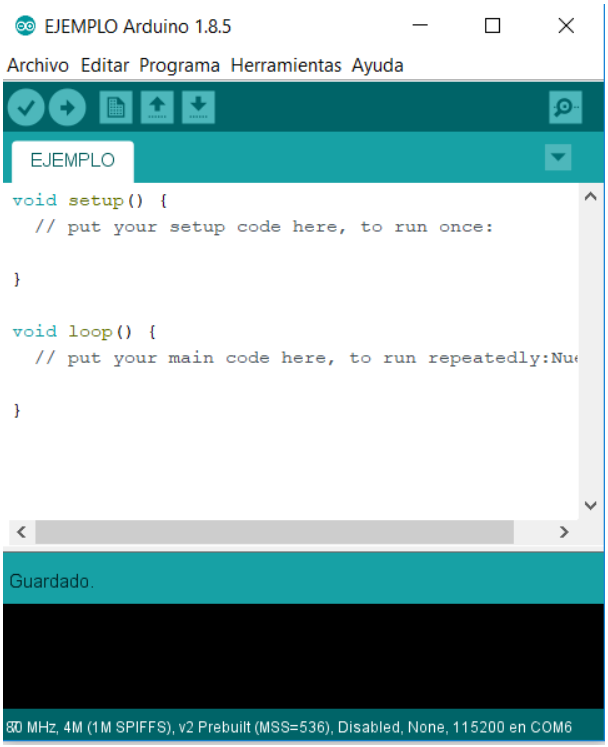


Figura 5.37. IDE Arduino.

En la IDE de Arduino se pueden distinguir cinco zonas distintas que podemos ver en la *figura 5.38* marcadas. La zona 1, la barra de menús, zona 2 barra de botones, zona 3 editor de código, zona 4 barra de mensajes y zona 5 barra de estado.

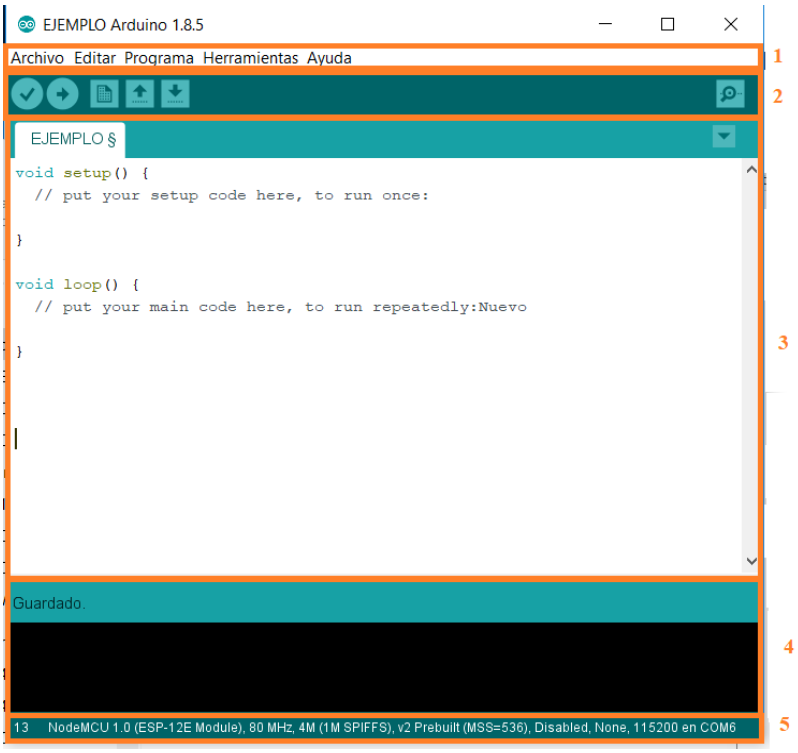


Figura 5.38. Zonas de trabajo de Arduino.

La zona 1 o barra de menús cuenta con 5 desplegables distintos, como se pude ver en la imagen, archivo, editar, programa, herramientas y ayuda. La pestaña de archivo tiene funciones como las de nuevo archivo, abrir proyectos, ejemplos que vienen definidos de serie, preferencias donde podemos modificar según interese como por ejemplo el idioma, la fuente, el tamaño de la fuente, en este menú no aparecen todos los ajustes que podemos modificar, si queremos modificar algo que no aparece, debemos ir a cambiarlo a fichero "preferences.txt". El menú editar permite acciones como cortar/pegar, hacer / deshacer, comentar / descomentar. El apartado de programa permite las acciones de compilar, subir incluir librerías y añadir ficheros. El apartado de herramientas permite dar auto formato al programa abrir el monitor y puerto serie y nos permite seleccionar la placa con a que se trabaje. Por último, el menú de ayuda permite el acceso a la página oficial de Arduino donde se pueden consultar tutoriales, artículos y ejemplos de ayuda, el acceso a estos documentos no es necesario ya que de descargan con el IDE de Arduino.

La zona 2 o zona de botones, *figura 5.39*, consta de seis botones distintos que explicaremos a continuación.

Figura 5.39. Zona 2 IDE Arduino.

El botón con la etiqueta 1 permite compilar el programa, es decir comprueba que no hay errores en el código. El dos es el botón para cargar el programa al microcontrolador, antes de subir el programa al microcontrolador vuelve a compilar. El botón 3 crea un nuevo programa en blanco. El cuatro permite abrir cualquier programa o ejemplo que tengamos

guardado previamente. Botón 5 para guardar el archivo y por último el Monitor serie que nos permite ver los datos e información cuando se lo indiquemos en el programa.

La zona 3 o editor de código es la zona donde se escriben los programas.

La zona 4 es la zona de barra de mensajes, donde aparece el estado del programa si ha sido guardado, si está compilando o si se ha subido al microcontrolador.

La zona 5 barra de estado muestra la línea en la que se encuentra el cursor escribiendo, la placa que se está utilizando y el puerto serie que en el que se está mostrando la información.

5.2. Configuración NodeMCU

Para preparar el entorno de Arduino, para el uso del módulo NodeMCU debemos seguir los siguientes pasos [16]:

Paso 1: Añadir la placa NodeMCU. Al instalar el IDE de Arduino, si busca la placa NodeMCU no aparece entre las disponibles, por ello hay que instalar las placas adicionales que cuentan con el módulo ESP8266 que es el que contiene nuestro microcontrolador. Para ello ir a la pestaña de Archivo → Preferencias, y en gestor de Urls adicionales de Tarjetas añadimos la dirección http://arduino.esp8266.com/stable/package_esp8266com_index.json y aceptamos.

Paso 2: añadir los drivers del ESP8266. Para ello accedemos a herramientas→ placa: Arduino UNO →Gestor de tarjetas. En el buscador introduzca "esp" y debería instalar la que se denomina "esp8266 by ESP8266 Community".

Paso 3: Seleccionar la placa con la que se vaya a trabajar. En el caso de este proyecto NodeMCU 1.0 (ESP-12E module).

Paso 4: Configuración de los parámetros de la placa.

- Placa:" NodeMCU 1.0 (ESP-12E module)"
- Flash Size: "4M (3M SPIFFS)"
- Debug port: "Disabled"
- Debug Level: "Ninguno"
- IwIP Variant: "v2 Prebuilt (MSS =536)"
- CPU Frequency: "80 MHz"
- Upload Speed: 9600. [7]

[7] Esta velocidad depende de la calidad del cable USB que tengamos.

5.3. Programación Lua

Lua es un lenguaje de programación de alto nivel, creado en 1993. Es un lenguaje basado en C, C++ y ANSI. Es un software libre y se puede utilizar en distintas plataformas. Este lenguaje de programación los datos al igual que ocurre en otros lenguajes de programación tienen un tipo, es decir pueden ser de tipo vector, valor lógico, entero, etc. pero en el caso de las variables no se tienen esa característica. Lua tienen una particularidad y es que lo vectores, cadenas o listas pueden ser representados utilizando la estructura en Lua denominada tabla [19].

5.4. Programación de comandos AT

Los comandos AT, abreviatura en ingles de "Attention". Son un conjunto de instrucciones codificadas que dan lugar a un lenguaje desarrollado por Dennis Hayes en 1977. Fue creado principalmente para comunicación con un modem, pero ha ido evolucionando y también se puede utilizar para comunicar con microcontroladores.

5.5. Comparación Arduino, Lua y comandos AT

Como se explica en los apartados anteriores ha podido observar existen tres formas distintas de programación del módulo ESP8266. Programar con comandos AT, limita mucho la programación, ya que no deja de ser una serie de sentencias que permiten comunicación, entonces queda elegir entre Arduino y Lua, seleccionamos Arduino porque existe una gran cantidad de librerías ya desarrolladas y la comunidad tan grande que utiliza Arduino y donde es posible encontrar ejemplos y ayuda a la hora de elaborar nuevos proyectos. Una de la desventaja con la que se puede encontrar el usuario son las posibles limitaciones de la propia IDE de Arduino.

6. MY APP INVENTOR 2

My App Inventor 2 es un entorno de desarrollo para creación de aplicaciones para el sistema operativo Android. Cuyo objetivo es facilitar la forma de crear aplicaciones. Esto lo consigue gracias al cambio en el lenguaje de programación que usa, se programa mediante bloques, y permite que sea mucho más sencillo la programación y cualquier persona sin necesidad de tener nivel elevado de conocimientos de programación puede usar esta herramienta. App inventor fue creado por el Instituto tecnológico de Massachusetts (MIT) como un proyecto dirigido por el profesor Hal Abelson junto con un equipo de personal y estudiantes. Es un entorno de desarrollo con más de 6 millones de usuarios que provienen de 195 países diferentes [20].

Lo primero que se puede observar al entrar en My App Inventor es la barra de menú principal, que se muestra en la *figura 6.40*. El primer despegable es de denominado "Projects". Aquí se puede acceder a los diferentes proyectos que tenga creados el usuario, crear un nuevo proyecto, importar o exportar un proyecto en formato .aia, guardar y borrar los proyectos. En la pestaña "Connect", aquí está la opción de conectar la aplicación al dispositivo mediante WiFi, USB o con el emulador y poder probarla sin necesidad de descargarla, también se puede resetear la conexión. El despegable "Built" podemos encontrar la posibilidad de generar un código QR para crear el archivo .apk para poder instalar la aplicación en el dispositivo, guardar el archivo QR en el ordenador o en Google Play. La pestaña "Help" aquí se pueden encontrar herramientas para ayudar al usuario a utilizar My App Inventor. La pestaña de "My projects" permite al usuario ver todas las aplicaciones que tiene creadas. El despegable "Guide" lleva al usuario a una página donde encontrará ejemplos y guías de uso. La pestaña "Report an Issue" como su nombre indica permite informar de un posible problema y consultar problemas antiguos. Por último, la pestaña de idioma que permite al usuario seleccionar el idioma para usar la herramienta.

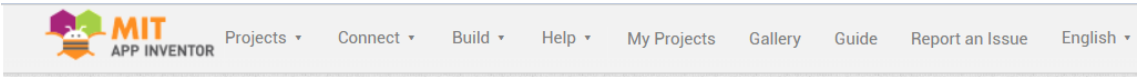


Figura 6.40. Barra de menú principal My App Inventor.

La interfaz está dividida en dos partes, la primera donde se lleva a cabo el diseño gráfico de la aplicación, decir donde se da forma a la parte que verá el usuario final de la aplicación y una segunda parte donde se lleva a cabo la programación de como actuará cada elemento de la aplicación, esta parte se programa mediante bloques lógicos.

1En la *figura 6.41*, se puede ver la parte de la interfaz gráfica de App inventor, en la zona "Viewer" podemos encontrar la pantalla del dispositivo donde haremos el diseño de la aplicación, para ello usaremos los componentes que aparecen en "Palette". Aquí encontramos elementos como botones, cajas de texto, imágenes, elementos de almacenaje, conectividad, mapas sensores, etc. En la parte de "Components" aparece un esquema de la estructura de la aplicación. En la sección "Properties" se pueden hacer cambios como el tamaño de la fuente, el color, situación de los elementos, etc.

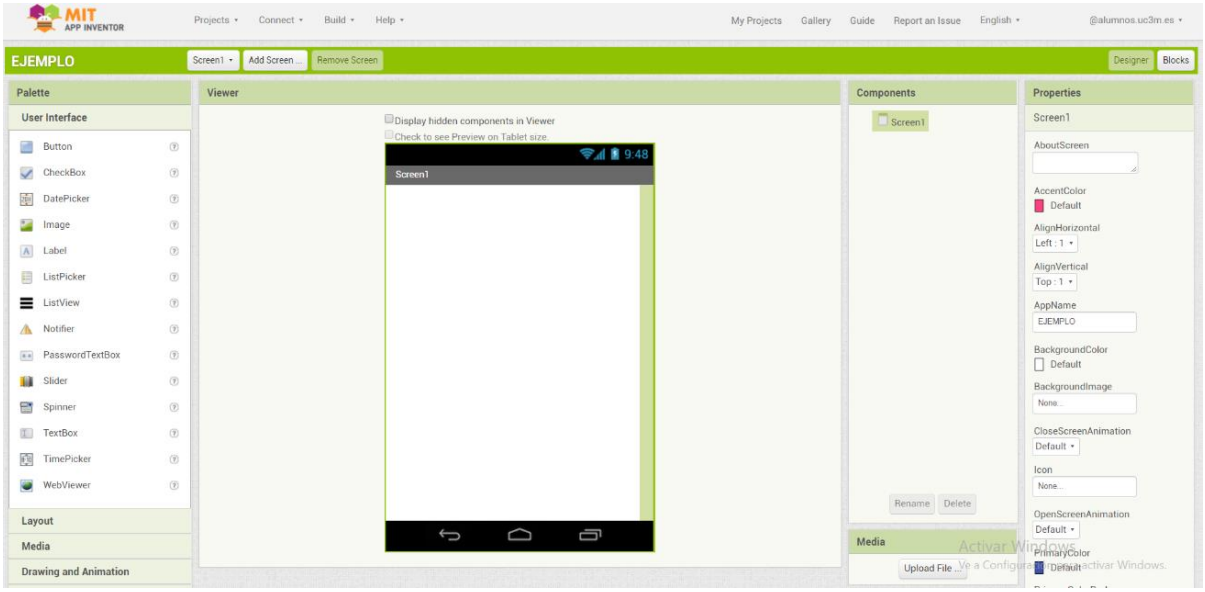


Figura 6.41. Interfaz gráfica de My App Inventor 2.

En la *figura 6.42*, se muestra la interfaz de programación, esta sección permite ir seleccionando los distintos bloques de programación lógica, para que la aplicación funcione como el diseñador desee. La mochila que se puede observar en la esquina superior derecha permite llevar los bloques de una pantalla a otra y de un proyecto a otro.

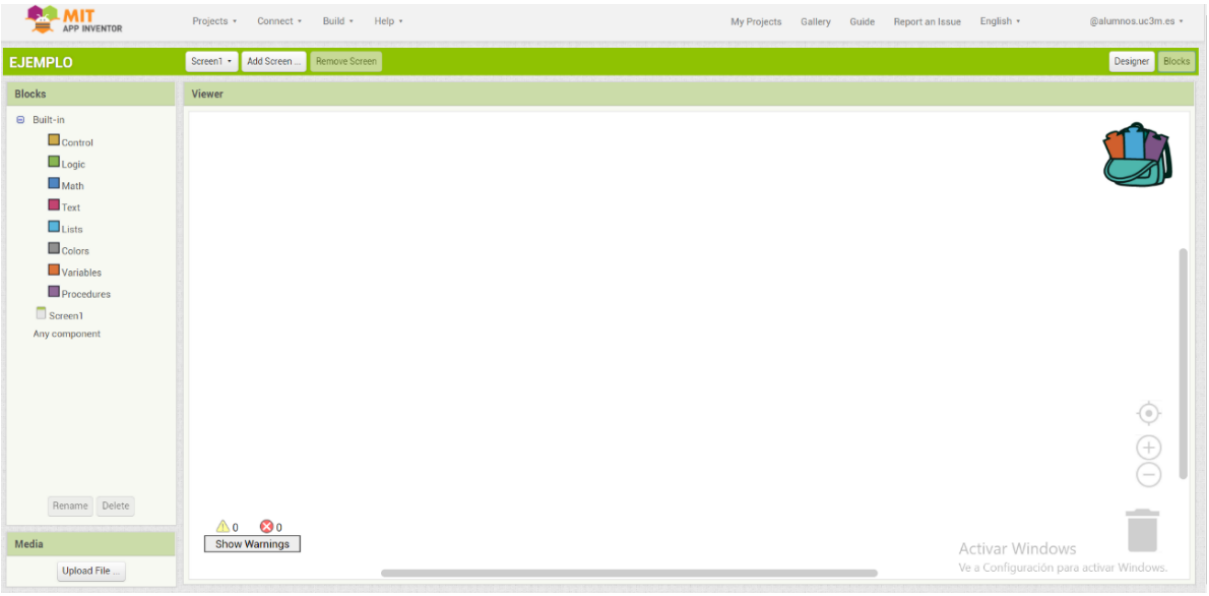


Figura 6.42. Interfaz diseño de My App Inventor 2.

Mientras se lleva a cabo el diseño de la aplicación, se puede ir comprobando el funcionamiento con la aplicación MIT AI2 Companion.

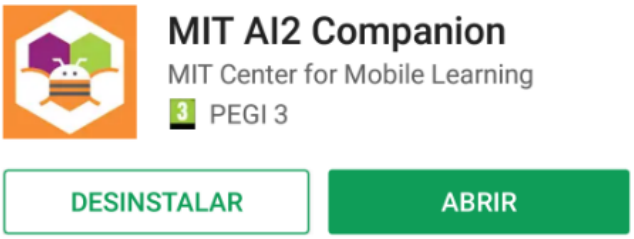


Figura 6.43. Miniatura de aplicación MIT AI2 Companion.

Una vez elaborado el diseño y la programación de la aplicación hay que crear un archivo .apk para instalarla en un dispositivo Android.

7. GOOGLE CHART

Google Chart es una aplicación de Google que da la posibilidad de visualizar gráficos en un sitio web. La aplicación proporciona diversidad de opciones para crear gráficos dependiendo de las necesidades de cada usuario. La forma más habitual de usar Google Chart es con JavaScript, los gráficos se representan usando la tecnología HTML5/ SVG para evitar problemas de incompatibilidades entre navegadores y entre Android e IOS [20].

Para la programación en Google Chart, se parte del ejemplo que se muestra en la *figura 7.44*.

```
<html>
<head>
  <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
  <script type="text/javascript">
    google.charts.load('current', {'packages':['corechart']});
    google.charts.setOnLoadCallback(drawChart);

    function drawChart() {
      var data = google.visualization.arrayToDataTable([
        ['Year', 'Sales', 'Expenses'],
        ['2004',  1000,      400],
        ['2005',  1170,      460],
        ['2006',  660,       1120],
        ['2007',  1030,      540]
      ]);

      var options = {
        title: 'Company Performance',
        curveType: 'function',
        legend: { position: 'bottom' }
      };

      var chart = new google.visualization.LineChart(document.getElementById('curve_chart'));

      chart.draw(data, options);
    }
  </script>
</head>
<body>
  <div id="curve_chart" style="width: 900px; height: 500px"></div>
</body>
</html>
```

Figura 7.44.Ejemplo de código de Google Chart [20].

El código de la figura anterior se crea una gráfica como la que aparece en la *figura 7.45*.

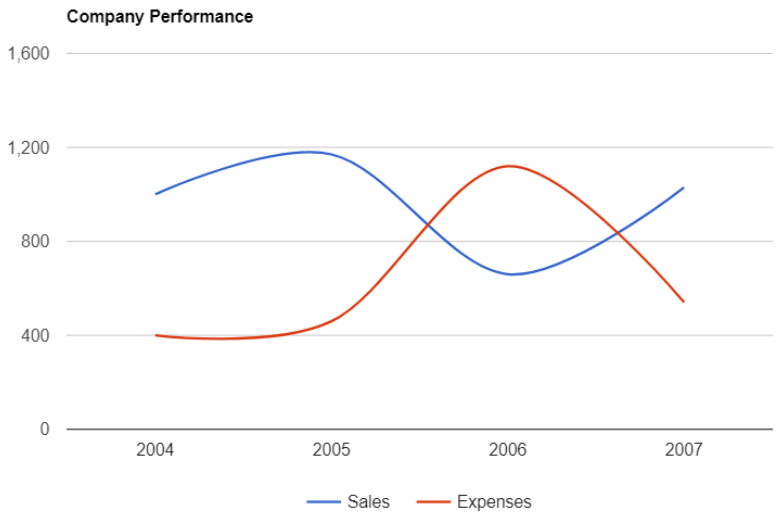


Figura 7.45. Visualización del ejemplo de Google Chart.

Para que el código grafique los datos de temperatura hay que seguir la siguiente estructura si quisiéramos una gráfica de dos variables “X” y “Y” los datos deberían tener la siguiente estructura: [['X', 'Y'],['0', 0],['1', 1],['2', 2],['3', 3]]. Cuya visualización sería la mostrada en la *figura 7.46*. En el apartado de diseño se explicará cómo crear esta estructura, para que la aplicación Android solo tenga que procesar la información.

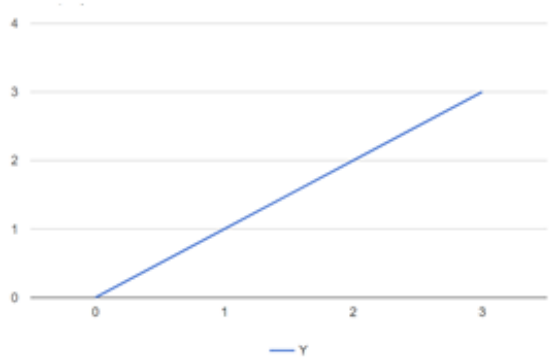


Figura 7.46. Ejemplo Google Chart.

8. DISEÑO

El sistema de medida de temperatura que se ha desarrollado está constituido por el sensor BME280, el microcontrolador NodeMCU y para poder ver el funcionamiento también será necesario acceso a una red WiFi y a un dispositivo Android. En la *figura 8.47* se muestra un esquema de cómo se han establecido las conexiones.

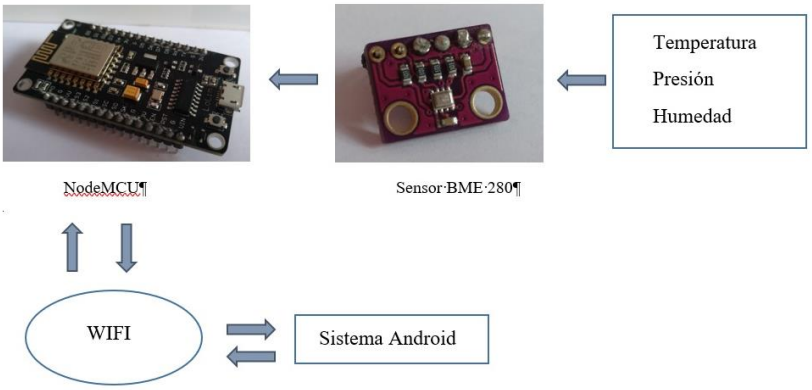


Figura 8.47. Esquema de conexiones sistema de medida.

Como se mencionó en *epígrafe 2.4* de medida tiene tres partes diferenciadas:

1. Adquisición de datos, esta etapa es realizada por el sensor.
2. Procesamiento de datos, el microcontrolador NodeMCU es el encargado de este proceso.
3. Distribución de datos, esta fase la realiza la aplicación diseñada mediante My App Inventor y Google Chart.

El la *figura 8.48*, se muestra como ha quedado el montaje del circuito final.

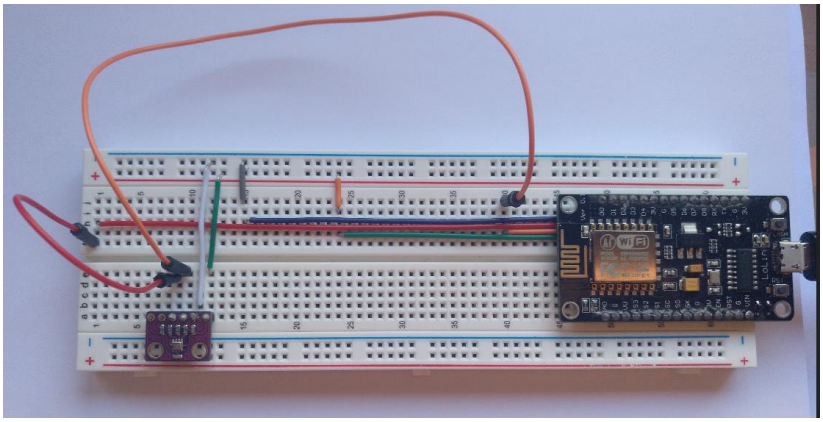


Figura 8.48. Montaje final.

8.1. Código en Arduino.

A continuación, se explica cada parte del código, que además se encuentra íntegro en el ANEXO B del presente documento.

El programa tiene cinco partes diferenciadas:

1. **Inicialización** de librerías, variables y constantes necesarias para la ejecución del programa.
   Lo primero que hace el programa es importar las librerías necesarias para el proyecto, en la *tabla 8.6*, encontramos un resumen de cada una de las utilizadas en el proyecto.

TABLA 8.6. LIBRERÍAS USADAS EN EL PROGRAMA.

Nombre de la librería	Descripción
ESP8266WiFi.h	Librería basada en SDK de ESP8266, que proporciona las rutas específicas WiFi de ESP8266 para conectarse a red.
ESP8266WebServer.h	Librería utilizada para crear un servidor web.
stdint.h	Librería para la programación integrada.
SparkFunBME280.h	Librería para uso del sensor BME280.
Wire	Librería de Arduino que permite la comunicación I2C.

A continuación, en la *figura 8.49*, se muestra como deberían inicializan las librerías, en el programa.

```
/*Librerías */

    #include <ESP8266WiFi.h>
    #include <ESP8266WebServer.h>

    #include <stdint.h>
    #include "SparkFunBME280.h"

    #include "Wire.h" //Librería para comunicación I2C
    //#include "SPI.h"//Librería para comunicación SPI
```

Figura 8.49. Librerías Arduino.

2. **Conexión a la red WiFi** En esta parte del código se lleva a cabo la conexión del ESP8266 con una red WiFi, para llevar a cabo posteriormente la conexión con la aplicación creada en My App inventor. En la *figura 8.50*, se muestran dos líneas importantes a la hora de ejecutar el programa y es imprescindible que cada usuario modifique, ya que si no el programa no funcionará. Son las siguientes constantes: "ssid" entre comillas debe aparecer el nombre de la red WiFi a la que vaya a conectarse y "password" entre comillas debe aparecer la contraseña de la red a la que se conecte. [9]

```
ESP8266WebServer server(80);

const char* ssid = "-------------"; // Red Wifi
const char* password = "-------------"; //Contraseña del Wifi
```

Figura 8.50. Constantes para a conexión a WiFi .

En la siguiente imagen, *figura 8.51*, se muestra la parte del código que establece la conexión WiFi.

```
//Conexión al Servidor//
WiFi.begin(ssid, password);
while(WiFi.status() !=WL_CONNECTED){
  delay(500);
  Serial.print(".");
  }
Serial.println("");
Serial.print("Conectado a ");
Serial.println(ssid);
Serial.print("IP address: ");// direccón IP  a la que se conecta
Serial.println(WiFi.localIP());
```

Figura 8.51. Conexión WiFi.

3. **Conexión del sensor/ lectura de datos.** El sensor se comunica con el microcontrolador mediante comunicación I^2C por eso hay que indicar a el microcontrolador el modo de funcionamiento y la dirección a la que se va a conectar el sensor, en este caso será a la 0x76, como se indica en la *figura 8.52.*

```
//Conexión I2C
  mySensor.settings.commInterface = I2C_MODE;
  mySensor.settings.I2CAddress = 0x76; //Dirección especifica para I2C puede ser 0x77 or 0x76
```

Figura 8.52. Conexión I2C.

En la *figura 8.53*, se muestra los diferentes modos de funcionamiento, que se pueden configurar en el sensor y cuales se han elegido para este proyecto. [22]

```
//Modos de operación Sensor BME 280

  //renMode can be:
  //  0, Sleep mode
  //  1 or 2, Forced mode
  //  3, Normal mode
  mySensor.settings.runMode = 3; //Normal mode

  //tStandby can be:
  //  0, 0.5ms
  //  1, 62.5ms
  //  2, 125ms
  //  3, 250ms
  //  4, 500ms
  //  5, 1000ms
  //  6, 10ms
  //  7, 20ms
  mySensor.settings.tStandby = 0;

  //filter can be off or number of FIR coefficients to use:
  //  0, filter off
  //  1, coefficients = 2
  //  2, coefficients = 4
  //  3, coefficients = 8
  //  4, coefficients = 16
  mySensor.settings.filter = 0;

  //tempOverSample can be:
  //  0, skipped
  //  1 through 5, oversampling *1, *2, *4, *8, *16 respectively
  mySensor.settings.tempOverSample = 1;

  //pressOverSample can be:
  //  0, skipped
  //  1 through 5, oversampling *1, *2, *4, *8, *16 respectively
    mySensor.settings.pressOverSample = 1;

  //humidOverSample can be:
  //  0, skipped
  //  1 through 5, oversampling *1, *2, *4, *8, *16 respectively
  mySensor.settings.humidOverSample = 1;
```

Figura 8.53. Configuración BME280.

En la *figura 8.54*, se muestra el bucle en el cual se inicializa el sensor y se comprueba su correcto funcionamiento.

```
// Inicialización del sensor

if (mySensor.begin())
Serial.print("BME280 se ha conectado correctamente");
else
Serial.print("La conexión de BME280 ha fallado, revise las conexiones");
```

Figura 8.54. Comprobación de conexión del sensor.

En la *figura 8.55* se muestra cómo se toman los valores del sensor de temperatura y el cálculo de la temperatura máxima y mínima.

```
//Lectura de valores

  arrayPot['X']={data};
 for (int i=0; i<X; i=i+1){

     data = (mySensor.readTempC()); //Lectura de valore sensor.
     //Serial.println(data);
     delay(T);
     arrayPot[i]=data;
     //Serial.println(arrayPot[i]);

//// Cálculo del máximo y mínimo temperatura
       //Cálculo Tmax
           if (arrayPot[i]> mayor){
             mayor=arrayPot[i];
            }
       //calculo Tmin
           if (arrayPot[i]< menor){
             menor=arrayPot[i];
             }

 }
```

Figura 8.55. Lectura sensor.

Las funciones que hay que usar para tomar las medias con el sensor BM280 son las que se indican en la *figura 8.56* [21].

```
//Funciones para lectura de diferentes variables.

    mySensor.readTempC(), 2; //Lectura de temperatura en grados centrigados.
    mySensor.readTempF(), 2;// Lectura de temperatura en grados Fahrenheit.

    mySensor.readFloatPressure(), 2; //Lectura de presión en pascales.
    mySensor.readFloatAltitudeMeters(),2 ; //Lecura de presión en metros.
    mySensor.readFloatAltitudeFeet(), 2;  //Lecura de presión en pies.
```

Figura 8.56. Funciones de lectura BME280

4. **Estructura de datos.** El programa en Arduino crea una estructura con los datos recogidos, para que Google Chart sea capaz de interpretar y poder graficar posteriormente. Esta estructura se podría crear en My App Inventor también, pero se ha decidido llevar a cabo toda la programación en Arduino para que la aplicación solo se encargue de graficar.

```
//Estructura para enviar datos a servidor

     appString += "[[\"X\",\"Y\"], ";

        arrayPot['X']={data};
        for (int i=0; i<X; i=i+1){
          appString += "[";
          appString += String(i);
          appString += ",";
          appString += String(arrayPot[i]);
          appString += "]";

           if (i<(X-1)){
              appString += ",";
            }
              appString += (" ");
       }

      appString += "]";

     //Serial.println (appString);

    server.send (200,"text/plain",(String)appString);
```

Figura 8.57. Estructura para My App Inventor

5. **Envío de datos al servidor.** En la *figura XX* se muestra cómo se realiza el envío de datos cuando la aplicación desarrollada lo solicita. Se muestra el ejemplo para descargar la temperatura máxima, pero tienen una estructura idéntica tanto para la descarga de temperatura mínima como para los datos de todo el día.

```
//Para descargar datos max temperatura//

     server.on("/Tmax",[](){

         server.send(200,"text/plain", (String)mayor);

     });
```

8.2. Código en Google Chart.

A continuación, en la *figura 8.58* se muestran los cambios que se han realizado en el código proporcionado por Google Chart para que tome los valores que recibe la aplicación desarrollada en My App Inventor.

```
<html>
  <head>
    <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
    <script type="text/javascript">
      google.charts.load('current', {'packages':['corechart']});
      google.charts.setOnLoadCallback(drawChart);
        d = window.AppInventor.getWebViewString();
        var array = JSON.parse(d);

      function drawChart() {
        var data = google.visualization.arrayToDataTable(array);


        var options = {
          title: 'Temperatura',
          hAxis: {title: 'Tiempo [s]' },
          vAxis: {title: 'Temperatura [ºC]'},
          legend: 'none'
        };

        var chart = new google.visualization.ScatterChart(document.getElementById('chart_div'));

        chart.draw(data, options);
      }
    </script>
  </head>
  <body>
    <div id="chart_div" style="width: 500px; height: 200px;"></div>
  </body>
</html>
```

Figura 8.58. Código Google Chart para graficar en My App Inventor 2.

8.3. Código con My App inventor.

En este apartado se va a explicar la programación que se ha realizado en el desarrollo de la aplicación.

El archivo creado con Google Chart, que se menciona en la *figura 8.58* hay que guardarlo como.html y cargarlo en My App inventor. Esto se debe hacer en la pestaña de “Designer” el apartado de media →Upload File y subimos el archivo creado, al crear una matriz de análisis d con JSON hay que marcar la pestaña de ShowListsAsJson en My App inventor, para que pueda funcionar [21], como se muestra en la *figura 8.59*. El archivo es guardado de forma automática en la siguiente dirección, file://mnt/scard/AppInventor/assets/nombredelarchivo.html cuando se está utilizando la aplicación de simulación, mientras que si ya se ha creado el archivo .apk la dirección se debe cambiar a file:///android_asset/nombredelarchivo.html.

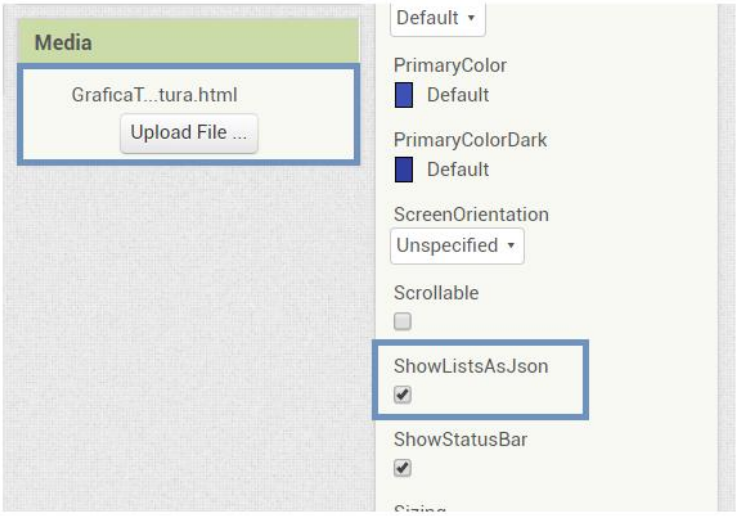


Figura 8.59. Ajustes My App Inventor.

De la parte de programación por bloques se muestra en la *figura 8.60* la configuración para la descarga de datos para graficar en Google Chart y en la *figura 8.61* se muestra la configuración del botón para la descarga de temperatura mínima, la del botón de temperatura máxima es exactamente igual.

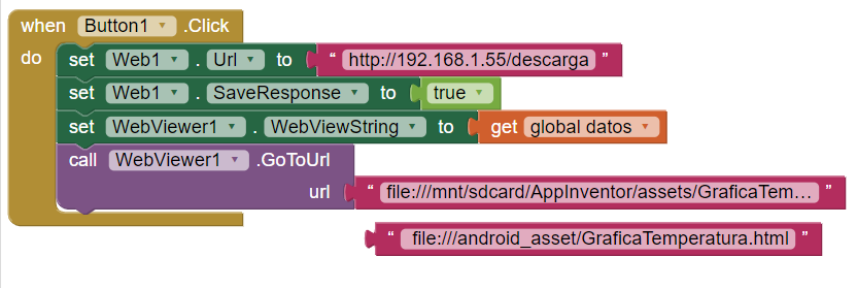


Figura 8.60. Configuración botón descargar datos.

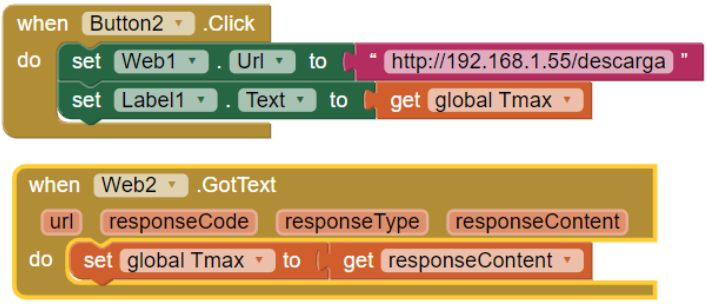


Figura 8.61. Configuración descarga temperatura máxima.

En la *figura 8.62*, a la izquierda se muestra como ha quedado el diseño de la aplicación, y a la izquierda aparece un ejemplo de resultados obtenidos.

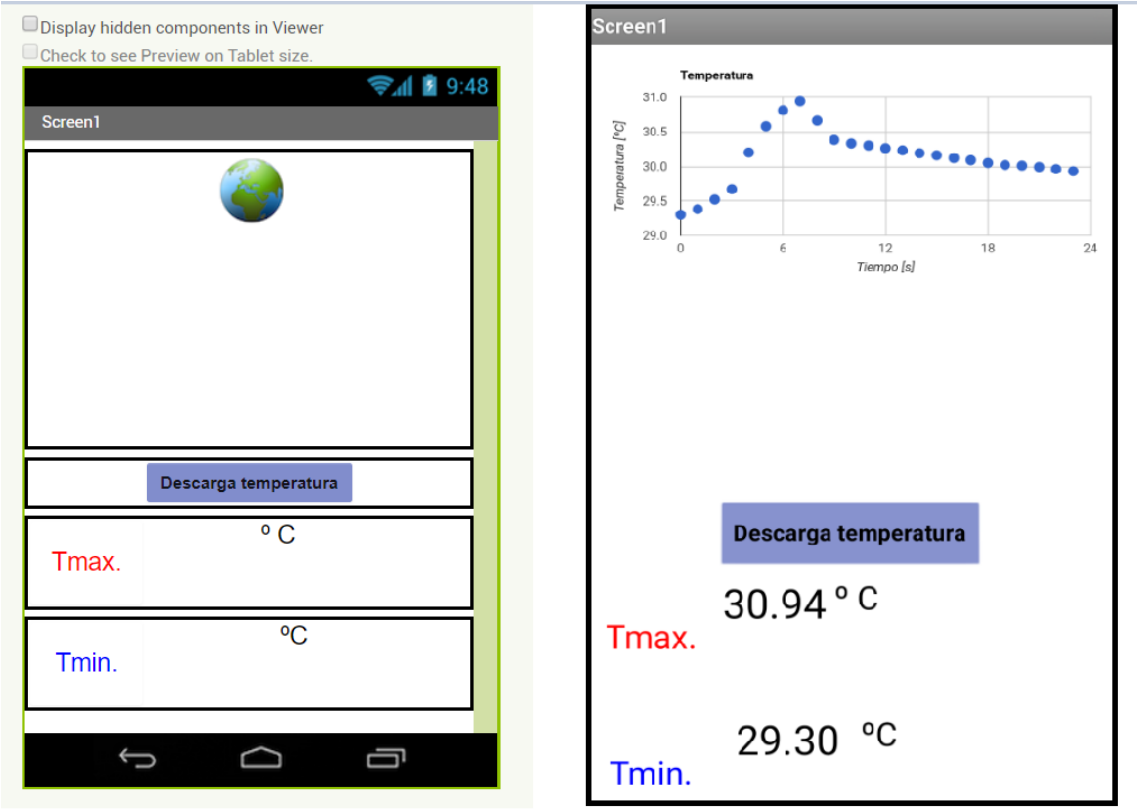


Figura 8.62. Esquema y ejemplo de App.

9. PRESUPUESTO Y PLANIFICACIÓN

9.1. Presupuesto

En el presente capítulo se desglosará el presupuesto empleado para la creación del prototipo.

Costes materiales

En las *tablas 9.7 y 9.8*, se detalla el presupuesto de cada uno de los elementos necesarios para la elaboración del sistema de medida.

TABLA 9.7 PRECIOS COSTE DE MATERIAL DEL SISTEMA DE MEDIDA.

Concepto	Precio unitario €	Cantidad	Total en €
Placa NodeMCU	9,00	1	9,00
Sensor BME280	7,00	1	7,00
Cables conexiones	3,00	1	3,00
Fuente de alimentación	2,00	1	2,00
Convertidor DC 9V 1	5,00	1	5,00
Cable USB	6,00	1	6,00
Placa protoboard	3,00	1	3,00
Total			**35,00**

TABLA 9.8. PRECIOS DE MATERIALES ADICIONALES PARA LA ELABORACIÓN DEL PROYECTO.

Concepto	Precio unitario €	Cantidad	Total en €
Soporte en impresión 3D	0,04€/cm^3	180 cm^3	7,20
Ordenador	800,00	1	800,00 €
Rúter	80,00	1	80,00
Dispositivo Android	180,00	1	180,00
Total			**1.067,20**

Presupuesto de mano de obra

En la *tabla 9.9*, se detalla el presupuesto de las horas necesarias para el desarrollo del prototipo.

TABLA 9.9. DESGLOSE DE PRECIOS DE MANO DE OBRA.

Concepto	Precio /hora €	Horas	Total en €
Ingeniero técnico	40,00	300	12.000,00
Total			**12.000,00**

Resumen presupuesto total.

En la tabla *9.10*, se muestra un resumen del coste total para la elaboración del proyecto.

TABLA 9.10. RESUMEN DE PRESUPUESTO.

Concepto	Coste total en €
Coste material total	1.102,20
Coste de mano de obra	12.000,00
Subtotal	13.102,20
Impuestos (21%)	2.751,47
Total	**15.853,67**

9.2. Presupuesto proyecto para la Comunidad de Madrid

Presupuesto para la instalación de una estación meteorológica en cada uno de los 179 municipios de la comunidad de Madrid, sería el que se detalla en las siguientes tablas.

El coste material sería el mismo que el calculado en las *tablas 9.7 y 9.8* en el caso de crear solo un dispositivo, pero en este caso para la elaboración de 179 estaciones meteorologías diferentes, también se crea una partida para repuestos por ello el presupuesto de costes materiales quedaría como aparece en la *tabla 9.11*.

TABLA 9.11. RESUMEN PRECIOS DE COSTE DE MATERIAL.

Concepto	Precio unitario €	Cantidad	Total en €
Estación meteorológica	1.102,20	179	197.293,80
Repuestos	1.102,20	10	11.022,00
Total			**208.315,80**

Por otra parte, en el apartado de mano de obra sería necesario tener en cuenta conceptos nuevos como un equipo de trabajo para la instalación de cada dispositivo en la ubicación deseada, se ha calculado una media de 3 horas para instalar cada uno de los dispositivos y otro equipo de mantenimiento para comprobar periódicamente que cada dispositivo funciona correctamente, en este caso se establece necesario 2 horas al mes para revisar cada dispositivo, en la *tabla 9.12* encontramos el desglose de cada una de las actividades.

TABLA 9.12. DESGLOSE DE PRECIOS DE MANO DE OBRA.

Concepto	Precio /hora €	Horas	Total, en €
Ingeniero técnico	40,00	300	12.000,00
Grupo de instalación	15,00	537	8.010,00
Mantenimiento	15,00	2.136,00	32.040,00
Total			**52.050,00**

Resumen presupuesto total.

En la *tabla 9.13*, se muestra un resumen de la elaboración del proyecto para instalar una estación meteorológica en cada uno de los 179 municipios de la Comunidad de Madrid.

TABLA 9.13. RESUMEN DE PRESUPUESTO PARA EL PROYECTO PARA LA COMUNIDAD DE MADRID.

Concepto	Coste total en €
Coste material total	208.315,80
Coste de mano de obra	52.050,00
Subtotal	260.365,80
Impuestos (21%)	54.676,82
Total	**315.042,62**

9.3. Planificación.

En la *figura 9.63* se muestra el tiempo que se ha planificado para cada una de las tareas en las que se ha dividido el trabajo. Cada una de las actividades corresponden a las siguientes tareas:

- La actividad A corresponde a la búsqueda de proyectos similares y búsqueda de posibles planteamientos para elaborar el trabajo.
- La actividad B es la etapa de compresión del funcionamiento del módulo ESP8266 y de la placa de desarrollo NodeMCU.
- La actividad C, aquí comprende el tiempo estimado necesario para la compresión del como elaborar la comunicación entre la placa de desarrollo NodeMCU con la red WiFi.
- La actividad D, consiste en la puesta en marcha del sensor y toma de valores.
- La actividad E, en esta etapa consiste en realizar pruebas y comprender el funcionamiento de las aplicaciones de Google Chart y My App Inventor 2.
- La actividad F, durante esta etapa se realiza la búsqueda de información para la elaboración de proyecto.
- La actividad G, elaboración de la memoria.
- La actividad H, es la revisión y corrección de la memoria.
- La actividad I, elaboración del PowerPoint.
- La actividad J, presentación y defensa del trabajo.

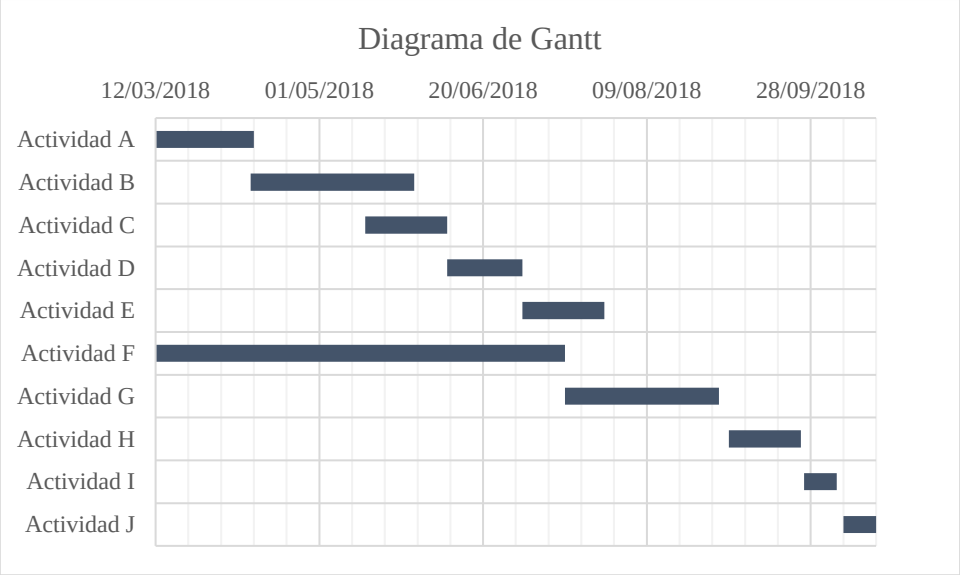


Figura 9.63. Diagrama de planificación.

10. CONCLUISIONES

10.1. Conclusiones

Hoy en día existe una necesidad de estar permanentemente contacto con el resto del mundo, el conocido mundo de Internet de las Cosas permite desarrollar e incrementar la conectividad de los objetos que nos rodean. Con el objetivo de explorar una de las múltiples ventajas del sistema de comunicaciones global en la actualidad, en este trabajo ha desarrollado un sistema de medida con control remoto a partir de Internet.

Se ha creado un prototipo de sistema de medida basado en el módulo ESP8266, junto con el sensor BME280 y herramientas de desarrollo.

La visualización de los datos tomados con el sensor BME 280 mediante I2C se hace mediante una gráfica creada con Google Chart. Para ello fue necesario establecer la comunicación entre My App Inventor mediante una estructura de datos JSON en forma de listas. Esta estructura la crea el ESP8266 como se muestra en el Capítulo 7. La utilización de Google Chart da gran versatilidad a la aplicación porque ajusta automáticamente los ejes de la gráfica a los datos disponibles.

Actualmente se puede controlar el tiempo que transcurre entre cada medición modificando el código creado con el IDE de Arduino.

El dispositivo que se ha desarrollado únicamente almacena el conjunto de datos que es enviado cada vez que la aplicación lo solicita. Es decir, muestra la información recogida en la aplicación creada. Esto se debe a que la memoria de la que dispone el microcontrolador es limitada, si el usuario necesita toda la información, habría que elaborar una base de datos y sería necesario añadir una ampliación de memoria, como podría ser una tarjeta SD. Al almacenar esta información sería necesario marcar cada medida obtenida con la fecha de hora y día, podría hacerse usando la librería Time.

Finalmente, la aplicación desarrollada ha demostrado ser capaz de cumplir con los propósitos planteados en el trabajo, facilitando la toma de medidas de temperatura y graficándolas mediante una aplicación para Android.

10.2. Líneas futuras de desarrollo

En este apartado se plantean posibles trabajos futuros para desarrollar o mejorar este proyecto.

Uno de los posibles caminos para seguir desarrollando el proyecto sería poder tener acceso a los datos de temperatura desde cualquier lugar del mundo con acceso a internet, y no tener la que estar conectado a la misma red WiFi para tener acceso.

El proyecto se ha planteado con un sensor de temperatura para una estación meteorológica, pero puede tener múltiples aplicaciones dependiendo de los sensores que se utilicen. Podría utilizarse un sensor para detectar niveles de contaminación, y crear un sistema de alarma para activar un protocolo de actuación.

Otra posible línea de desarrollo de este proyecto es el enfoque hacia la domótica. Podría ser la toma de medida de temperatura de una habitación y conectar un actuador para el control de la caldera y de aire acondicionado, estableciendo unas condiciones de umbral de bienestar, para que accione el aire acondicionado o caldera.

BIBLIOGRAFÍA

[1] Cisco, «CiscoIBSG,» EEUU, 2010.

[2] F. A. Fermín Pérez y J. L. Guerra Guerra, «Internet de las Cosas,» *Perspectivas,* vol. 10, nº 11, pp. 45-49, 11 07 2015.

[3] P. Ballarín Usieto, «Govertis,» Govertis, 9 04 2016. [En línea]. Available: https://www.govertis.com/gestionar-la-st-de-las-cosas. [Último acceso: 22 09 2018].

[4] Bosch Sensortec, «BME280 Combined humudity and pressure sensor,» Bosch Sensortec GmbH, Germany, 2014.

[5] J. Santaella, «Internet de los objetos: un futuro muy cercano,» *BIT*, nº 178, p. 58, 5 10 2010.

[6] K. Ashton, «RFID Journal,» Haycco, 22 06 2009. [En línea]. Available: https://www.rfidjournal.com/articles/view?4986. [Último acceso: 30 08 2018].

[7] O. Moreno Sanchez, «IoT con servicios en la nube de Microsoft Azure: diseñi y despliegue de una arquitectura IoT para el análisis de datos en tiempo real,» Tesis (Master), E.T.S.I. de Sistemas Informáticos (UPM), Madrid, 2017.

[8] J. Salazar, Redes Inalámbricas, República Checa: České vysoké učení technické v Praze.

[9] I. Grokhotkov, «ESP8266 Arduino Core,» GitHub, 2017. [En línea]. Available: https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html. [Último acceso: 19 09 2018].

[10] Ó. Torrente Artero , El mundo Genuino-Arduino. Curso práctico de formación., Madrid: RC Libros, 2016.

[11] M. A. Pérez García, J. C. Álvarez antón , J. C. Campo Rodriguez, F. J. Ferrero Martín y G. J. Grillo Ortega , Instrumentación Electrónica, Madrid: Thomson, 2004.

[12] A. Serna Ruiz, F. A. Ros García y J. C. Rico Noguera, Guía Práctica de Sensores, España: Creaciones Copyright, 2010.

[13] «Arduino,» [En línea]. Available: https://www.arduino.cc. [Último acceso: 15 08 2018].

[14] GNUIEN, «GNU Operating System,» Free Sofware Foundation, 13 Febrero 2001. [En línea]. Available: https://www.gnu.org. [Último acceso: 16 08 2018].

[15] L. d. Valle, «ProgramarFacil,» [En línea]. Available: https://programarfacil.com. [Último acceso: 20 08 2018].

[16] E. I. Team, «ESP8266EX Datasheet,» Espressif Inc. , 2018.

[17] M. Stör, «my2cents,» 28 Septiembre 2015. [En línea]. Available: https://frightanic.com. [Último acceso: 21 Agosto 2018].

[18] R. Lerusalimschy , L. Henrique de Figueiredo y W. Celes, «Manual de Referencia de Lua 5.1,» LUA, 2008. [En línea]. Available: https://www.lua.org/manual/5.1/es/manual.html. [Último acceso: 30 08 2018].

[19] Massachusetts Institute of Technology, «Mit App Inventor,» Massachusetts Institute of Technology, 2012. [En línea]. Available: http://appinventor.mit.edu. [Último acceso: 2018 08 25].

[20] Google, «Google Charts,» Google , [En línea]. Available: https://developers.google.com/chart/. [Último acceso: 2018 08 27].

[21] Sparkfun, «Sparkfun,» Sparkfun, [En línea]. Available: https://learn.sparkfun.com/tutorials/sparkfun-bme280-breakout-hookup-guide. [Último acceso: 07 09 2018].

[22] G. Robles Muñoz, «Electrica UC3M,» 17 05 2018. [En línea]. Available: http://electrica.uc3m.es/grobles/blog/?p=1348. [Último acceso: 12 06 2018].

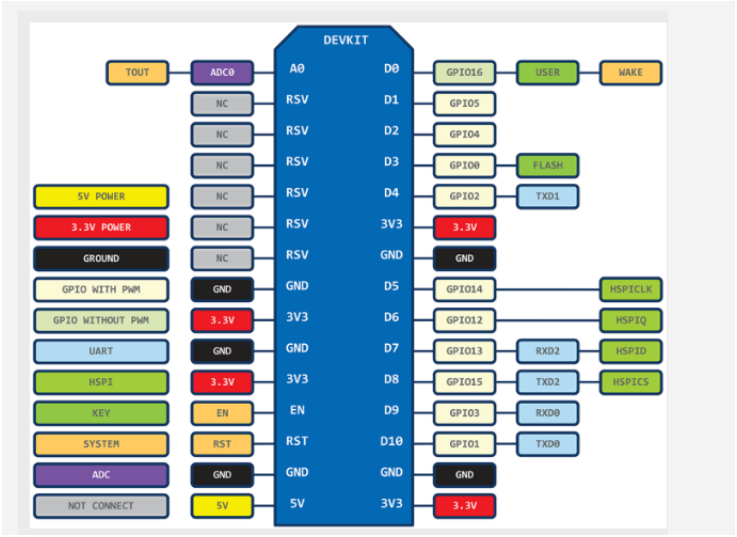
[23] L. d. Física, Guía práctica para la realización de la medida y cálculo de errores, Leganés, 1998.

[24] Aprendiendo Arduino, «Aprendiendo Arduino,» Aprendiendo Arduino, 13 09 2017. [En línea]. Available: https://aprendiendoarduino.wordpress.com/tag/comandos-at/. [Último acceso: 20 09 2018].

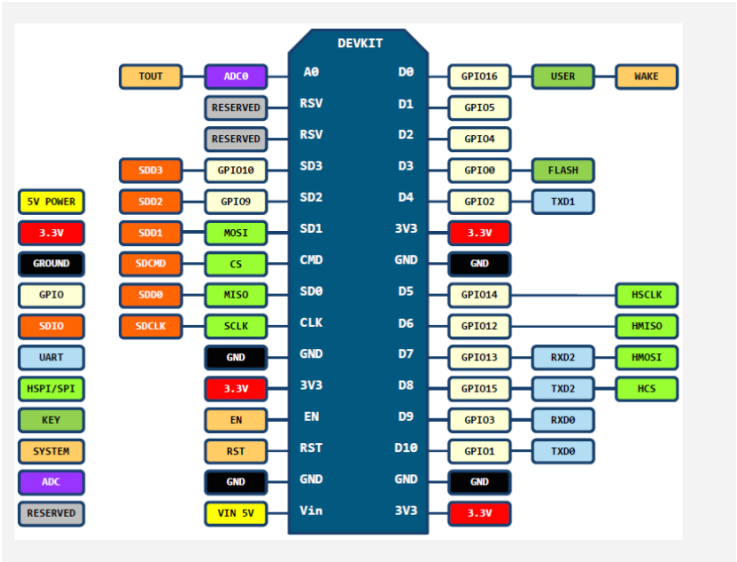
ANEXO A. NODEMCU

A continuación, se muestran los pinout de las tres versiones de NodeMCU [11]

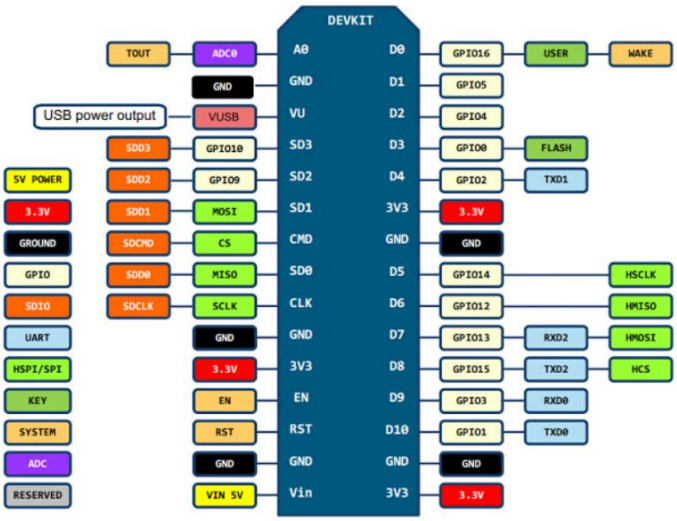
1ª Generación / V 0.9/ V1



2ª Generación/V 1.0 / V2



2ª Generación / V 1.0/ V3



ANEXO B. CÓDIGO FINAL DEL PROGRAMA

```
////El siguiente programa crea un servidor para enviar los datos de
temperatura del sensor BME280

  /*Librerías */

      #include <ESP8266WiFi.h>
      #include <ESP8266WebServer.h>

      #include <stdint.h>
      #include "SparkFunBME280.h"

      #include "Wire.h" //Librería para comunicación I2C

//////////////////////////////////////////////////////////////
////////Global sensor object//////////////////////////////////
//////////////////////////////////////////////////////////////

      BME280 mySensor;

//////////////////////////////////////////////////////////////
////////Conexión de Servidor WEB//////////////////////////////
//////////////////////////////////////////////////////////////


ESP8266WebServer server(80); //Puerto de comunicación del servidor.


//    const char* ssid = "-------------"; // Red Wifi
//    const char* password = "-------------"; //Contraseña del Wifi

const char* ssid = "MiFibra-1137"; // Red Wifi Casa
const char* password = "fWcziq5c"; //Contraseña del Wifi Casa




//////////////////////////////////////////////////////////////
////////Variables para el programa////////////////////////////
//////////////////////////////////////////////////////////////
               int cont=0 ;
               int Y=2;
               const float X=24;
               double arrayPot['X'];
                int i=0;
                int j=0;
               double data;

            String appString;

double mayor=0.00;
double menor=1000000.00;

int T=100; //Variable para determinar cada cuanto tiempo toma un
valor.

void handleRoot(){
```

```
  //server.send(200,"text/plain","Hello Teresa from esp8266, usted se
ha conectado");
  //server.send(200,"text/plain",(String)data);
}

void setup() {

Serial.begin(57600);

////////////////////////////////////////////////////////////////////
////////Conexión al Servidor////////////////////////////////////////
////////////////////////////////////////////////////////////////////

     WiFi.begin(ssid, password);

      while(WiFi.status() !=WL_CONNECTED){
        delay(500);
        Serial.print(".");
        }

      Serial.println("");
      Serial.print("Conectado a ");
      Serial.println(ssid);
      Serial.print("IP address: ");// direccón IP  a la que se conecta
      Serial.println(WiFi.localIP());

     server.on("/",handleRoot);//La segunda indica eñ bucle que va a
mostrar si pone handleRoot se muestra lo que aparece en ese bucle.
     server.begin();
     Serial.println("HTTP server started");

////////////////////////////////////////////////////////////////////
////////Descarga de  datos /////////////////////////////////////////
////////////////////////////////////////////////////////////////////


     server.on("/descarga",[](){

          //Estructura para enviar datos a servidor

            appString += "[[\"X\",\"Y\"], ";
            arrayPot['X']={data};

                for (int i=0; i<X; i=i+1){
                  appString += "[";
                  appString += String(i);
                  appString += ",";
                  appString += String(arrayPot[i]);
                  appString += "]";
                    if (i<(X-1)){
                      appString += ",";
                    }
                   appString += (" ");
                }
            appString += "]";

       server.send (200,"text/plain",(String)appString); //envío de
datos

     });
```

```
      //Para descargar datos maxima temperatura//

           server.on("/Tmax",[](){
               server.send(200,"text/plain", (String)mayor);
           });

        //Para descargar datos minima temperatura//

             server.on("/Tmin",[](){

                 server.send(200,"text/plain", (String)menor);

              });

////////////////////////////////////////////////////////////////
////////Conexión Sensor I2C ///////////////////////////////////
////////////////////////////////////////////////////////////////

   mySensor.settings.commInterface = I2C_MODE;
   mySensor.settings.I2CAddress = 0x76; //Dirección especifica para
I2C puede ser 0x77 or 0x76

      //Modos de operación Sensor BME 280

        //renMode can be:
        //  0, Sleep mode
        //  1 or 2, Forced mode
        //  3, Normal mode
        mySensor.settings.runMode = 3; //Normal mode

        //tStandby can be:
        //  0, 0.5ms
        //  1, 62.5ms
        //  2, 125ms
        //  3, 250ms
        //  4, 500ms
        //  5, 1000ms
        //  6, 10ms
        //  7, 20ms
        mySensor.settings.tStandby = 0;

        //filter can be off or number of FIR coefficients to use:
        //  0, filter off
        //  1, coefficients = 2
        //  2, coefficients = 4
        //  3, coefficients = 8
        //  4, coefficients = 16
        mySensor.settings.filter = 0;

        //tempOverSample can be:
        //  0, skipped
        //  1 through 5, oversampling *1, *2, *4, *8, *16 respectively
        mySensor.settings.tempOverSample = 1;

        //pressOverSample can be:
        //  0, skipped
        //  1 through 5, oversampling *1, *2, *4, *8, *16 respectively
          mySensor.settings.pressOverSample = 1;

        //humidOverSample can be:
```

```
        //  0, skipped
        //  1 through 5, oversampling *1, *2, *4, *8, *16 respectively
        mySensor.settings.humidOverSample = 1;


     delay(10); //Tiempo de espera para que el sensor tenga tiempo de
iniciarse.

  // Inicialización del sensor

  if (mySensor.begin(), HEX)
      Serial.println("BME280 se ha conectado correctamente");
  else
      Serial.println("La conexión de BME280 ha fallado, revise las
conexiones");

}

void loop() {

//Lectura de valores

   arrayPot['X']={data};
      for (int i=0; i<X; i=i+1){

      data = (mySensor.readTempC()); //Lectura de valore sensor.
      //Serial.println(data);
      delay(T);
      arrayPot[i]=data;
      //Serial.println(arrayPot[i]);

        // Cálculo del máximo y mínimo temperatura
            //Cálculo Tmax
                if (arrayPot[i]> mayor){
                  mayor=arrayPot[i];
                 }
            //calculo Tmin
                if (arrayPot[i]< menor){
                  menor=arrayPot[i];
                  }

       }


  delay(1000);

server.handleClient();

}
```